SPECTRUM TOOL

Programmi di utilità, grafica e gioco

Roberto Rigo

GRUPPO EDITORIALE JACKSON

SPECTRUM TOOL

Programmi di utilità, grafica e gioco

di
Roberto Rigo

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

SOMMARIO

PARTE I

UTILITÀ

PREFAZIONE

Nella realizzazione di questo libro ho voluto mettere in risalto due cose:

— la prima, la più importante, è mostrare come sia possibile realizzare programmi efficienti utilizzando due tecniche di programmazione e facendole interagire: il BASIC e il LINGUAGGIO MACCHINA (L/M).
  Il perchè di questa scelta che comporta qualche compromesso, è molto semplice.
  Il BASIC è un buon linguaggio ma ha lo svantaggio di essere molto lento.
  Il LINGUAGGIO MACCHINA è molto veloce ma è abbastanza complicato programmarlo.
  Poter utilizzare l'uno e l'altro porta ad una via di mezzo tra ciò che è la praticità d'uso e la velocità di esecuzione di un programma. In generale ho voluto destinare al BASIC la gestione dei calcoli, degli input e output, lasciando al L/M l'esecuzione di brevi ma veloci subroutine di varia utilità.
— La seconda cosa è strettamente collegata alla prima.
  A volte non ci si accontenta di acquistare il software già pronto, bensì lo si vuole creare. È qui che si rivela la necessità di saper ben programmare in BASIC, ma anche in L/M.

Sono del parere che nessuno saprà mai sfruttare a fondo il proprio computer se non sa almeno un poco dialogare con il microprocessore che esso contiene.

Per concludere:

Le vie scelte nello sviluppare il software proposto possono essere ampiamente criticate. Ci sono infatti molti modi per affrontare e risolvere i vari pro-

blemi. Io ne propongo uno, e son sicuro che sulla base di quanto si può ricavare da questo libro, ciascuno saprà elaborare il proprio software adattandolo alle proprie necessità nel migliore dei modi.

Norme per l'introduzione dei listati

Ora che il lavoro è finito e che ho in mano una pila di listati, mi accorgo della necessità di dare qualche informazione su come digitarli nel computer.

La maggior parte dei programmi sono stati fatti per poter girare su tutte le versioni dello Spectrum e perciò anche nelle due configurazioni di memoria, 16 e 48 Kbytes.

Molti di questi sono lunghi e a volte si dividono in due parti. Quando questo si verifica è perchè i due listati nella memoria di un 16k non ci stanno.

È consigliabile perciò seguire le istruzioni al seguito di ogni programma.

In generale uno dei due listati costituisce la parte in LINGUAGGIO MACCHINA (L/M) codificata in esadecimale nelle linee di DATA, esempio:

```
9000 DATA "3E","02","CD","01","16", ecc...
```

In questi casi occorre fare molta attenzione nel digitare i numerosi codici e, fatto molto importante, essi devono essere introdotti col cursore nel modo CAPS SHIFT (o CAPS LOCK), cioè in maiuscolo.

Sempre prima di far girare il programma, bisogna salvarlo su nastro. Le routine in L/M sono delle belle cose ma un solo bit fuori posto manda in tilt il computer.

Con un po' di esperienza sarà facile ridurre i due generici listati in unico blocco, ad esempio accodando al programma principale su nastro, la parte in L/M richiamabile poi con un LOAD "nome del file" CODE xxxx dopo aver eseguito un CLEAR xxxx–1 dove xxxx è l'indirizzo da cui iniziare a memorizzare l'L/M.

Nei vari programmi di questo libro e particolarmente nei giochi, si fa largo uso di caratteri grafici. Questi nei listati sono stampati usando il carattere A.S.C.I.I associato. La stampante usata non è infatti quella di produzione Sinclair e per motivi legati alla grafica ho ritenuto fosse più semplice seguire questa strada. È stato invece possibile stampare i caratteri grafici visibili sui tasti numerici dello Spectrum avendoli resi compatibili con la matrice utilizzata dalla stampante.

Struttura dei programmi

Come già spiegato in precedenza, questo libro mostra uno dei modi per programmare facendo agire fra loro il linguaggio BASIC e l'L/M. È chiaro che il metodo proposto non ha la pretesa di essere l'assoluto e comunque il più corretto.

Ci sono modi più efficienti per programmare e, prima ancora, di affrontare un problema.

I programmi in questo libro sono per il novanta per cento di mia produzione mentre gli altri sono il risultato di una rielaborazione (in meglio spero) di ciò che già c'era di interessante.

La maggior parte si compone di due parti:

1) il programma vero e proprio,

2) il listato assembler delle routines di cui esso si serve.

Ogni programma già contiene la codifica di tali routines in L/M in un certo numero di linee di DATA ma ho ritenuto fosse cosa utile dare anche il listato assembler di ciascuna di esse perchè fosse possibile con un po' di pazienza apportare le eventuali necessarie modifiche.

Allo scopo gli assembler sono commentati per gruppi di linee dando ove possibile la descrizione operativa delle routines che risiedono in ROM.

In generale i programmi in L/M sono stati fatti per risiedere in RAM a partire dall'indirizzo 32000 per dare la possibilità di utilizzo anche a coloro che possiedono uno Spectrum 16k.

Per certe applicazioni ciò non è stato possibile per cui solo l'utente di un 48k potrà servirsene.

La maggior parte delle routine sono rilocabili. In linea di principio ciò significa che è possibile memorizzarle in ogni zona della memoria senza pregiudicarne il funzionamento.

Nello Spectrum ciò non è sempre vero. Ci sono zone di RAM utilizzate dal BASIC in vari modi, delle quali risulta pericoloso servirsi. A volte è la routine stessa che utilizzerà la zona di RAM nella quale l'abbiamo incautamente posta.

Tutti i listati sono il prodotto di una stampante e interfaccia parallela, dopo aver provato ciascun programma su uno Spectrum ISSUE TWO.

Credo che non vi siano problemi di sorta con gli Spectrum della serie suc-

cessiva e comunque con le varie interfacce ad essi collegate, MICRODRIVE compresi.

Per concludere desidero ringraziare il Sig. Gian Maria VITTADINI per la pazienza dimostrata nel darmi consigli per la stesura del libro e soprattutto per la collaborazione prestata nel redigere i vari programmi di gioco e la loro composizione grafica.

HEX LOADER

Questo è un semplice programma che serve per poter memorizzare in RAM valori numerici posti nelle linee di DATA.

Perciò si userà spesso in quei programmi del libro che utilizzeranno routine in L/M, o comunque ove sia necessario porre in zone di RAM dei numeri ben determinati. Questi valori per poter essere manipolati dal programma in oggetto, devono essere inseriti come numeri esadecimali e visti come una stringa alfanumerica di due digit. La generica linea di DATA avrà questa forma:

1000 DATA "FO","1E","08", ecc...

Il valore numerico decimale viene estratto da questa stringa con poche e semplici manipolazioni e quindi messo in RAM (istruzioni POKE). L'elaborazione prosegue fino a che la stringa letta contiene i due asterischi "**" che rappresentano il marker di fine dati.

È necessario abbassare la RAMTOP se si desidera memorizzare questi dati in zone di memoria normalmente accessibili al BASIC. Ciò viene fatto con un CLEAR nnnn, essendo nnnn+1 la locazione iniziale ove memorizzare i dati.

```
9000 REM  HEX-CODE Caricatore
9005 REM
9010 REM Introdurre i dati HEX
9015 REM  con il CURSORE  nel
9020 REM    modo CAPS-LOCK
9025          POKE 23658,8
9030 CLEAR 31999: LET a=32000
```

```
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM  SUB NOME DEL PROGR.
9095 REM
9100 DATA "**"
```

DEC LOADER

Ovviamente esiste una maniera più semplice per memorizzare gli stessi dati. Basta cioè inserire i valori decimali nelle linee di DATA, leggerli e senza elaborazioni metterli in RAM. In tal caso la generica linea di DATA avrà la seguente forma:

9100 DATA 240,30,8, ecc...

Tuttavia mi permetto di consigliare un'ulteriore modifica:

9100 DATA "240", "30", "8", ecc...

Questa modifica appesantisce il lavoro del programmatore, ma occupa meno spazio in memoria e poi è perfettamente compatibile con il programma più avanti presentato (HEX-DEC DATA FORMAT).

```
9000 REM  DEC-CODE Caricatore
9005 REM
9010 REM Introdurre i dati DEC
9015 REM  tra gli apici ""
9020 REM  Es. "255","10",ecc.
```

```
9025 REM
9030 CLEAR 31999: LET a=32000
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 REM
9050 REM
9055 LET v=VAL a$: POKE a,v
9060 REM
9065 REM
9070 LET a=a+1: GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM  SUB NOME DEL PROGR.
9095 REM
9100 DATA "**"
```

NUMERO IN MEMORIA

Come si vede in figura, porre il numero fra gli apici fa risparmiare 4 byte per ogni numero che si considera.

Ciò avviene perchè ogni costante numerica viene posta in memoria come un insieme di caratteri ASCII seguiti dal byte 0E (in esadecimale) che indica al calcolatore che i 5 byte successivi sono la rappresentazione floating-point o intera del numero in questione.

Questo non accade se il numero viene posto fra gli apici, esempio: "240", perchè esso viene visto come stringa alfanumerica.

```
      Esempio di come e'
     memorizzata  una linea
       di DATA in memoria

9100 DATA   240  /   9100 DATA  "240"

  9100=238C  Numero di linea
         XX  Due byte di
         XX  Lunghezza linea
  DATA=  E4  Codice DATA
```

```
32  ---> 2                    22  ---> "
34  ---> 4                    32  ---> 2
30  ---> 0                    34  ---> 4
0E  ---> PREFISSO #           30  ---> 0
00  ---> 00                   22  ---> "
00  ---> 00
00  ---> 00      LO
F0  ---> 240     HI
00  ---> 00
```

HEX-DEC DATA FORMAT

Questo è un breve programmino che si avvale di una piccola routine in linguaggio macchina. A che cosa serve? È semplice. Qualsiasi dato numerico compreso tra 0 e 255, un byte, viene prelevato dalla memoria all'indirizzo specificato, convertito opzionalmente in esadecimale e ficcato in una linea di DATA che si andrà a posizionare in coda al programma.

L'utilità di questo programma sarà più evidente a coloro che sono abituati ad inserire nei propri listati Basic un certo numero di dati numerici: per esempio una routine in L/M, come in questo caso, oppure dati che definiscono un nuovo carattere grafico.

C'è poco da dire sul programma in sè. Penso che sia chiaro e comprensibile a tutti. Bisogna però fare attenzione a una cosa: la variabile "r" stabilisce il numero di dati che dovranno comparire in ogni linea. È bene non esagerare con questo valore per non correre il rischio di bloccare il sistema. La routine in L/M infatti adopera il buffer della stampante come memoria temporanea per costruire la linea Basic. Questo buffer è limitato a 256 byte quindi è molto facile saturarlo.

Il primo comando da dare una volta caricato il programma è RUN 9000 per permettere di caricare il L/M. Si potranno poi cancellare le linee di data che non servono più e iniziare a riprogrammarne di nuove.

Come sempre invito a dare un'occhiata al listato assembler per cercare di capire come il tutto funziona.

```
8000 REM
8005 REM      DATA  LINE  FORMAT
8010 REM
8015           POKE 23658,8
```

```
8020 LET data=32000: LET ok=1
8025 LET r=4: LET c=23296: LET fl=0
8030 LET p=c+5: LET scroll=23692
8035 LET s=0: POKE p-1,228: CLS
8040 INPUT  FLASH 1;"Inizio DATI ? ";i
8045 INPUT  FLASH 1;"FINE DATI ? ";f
8050 INPUT  FLASH 1;"Formato HEX/DEC ? "
; LINE a$
8055 IF a$(1)="H" THEN  LET fl=1
8060 REM
8065 REM     INIZIO  ROUTINE
8070 REM
8075 FOR j=i TO f: LET n=PEEK j
8080 GO SUB 8115: NEXT j
8085 IF ok THEN  GO SUB 8280
8090            POKE 23658,0
8095 BEEP .1,20: STOP
8100 REM
8105 REM           POKE "xx",
8110 REM
8115 POKE p,34: LET p=p+1
8120 IF fl THEN  GO TO 8145
8125 LET n$=STR$ n
8130 FOR z=1 TO LEN n$
8135 POKE p,CODE n$(z)
8140 LET p=p+1: NEXT z: GO TO 8160
8145 GO SUB 8250
8150 POKE p,h+48+7*(h>9): LET p=p+1
8155 POKE p,l+48+7*(l>9): LET p=p+1
8160 POKE p,34: LET p=p+1
8165 POKE p,44
8170 LET p=p+1: LET s=s+1
8175 IF j<>f AND s<r THEN  RETURN
8180 IF j=f AND s<r THEN  GO TO 8280
8185 POKE (p-1),13: LET s=0
8190 REM
8195 REM     SUB  # di LINEA
8200 REM
8205 LET n=p-c-4: GO SUB 8245
8210 POKE c+2,l: POKE c+3,h
8215 LET p=c+5: POKE scroll,-1
8220 RANDOMIZE USR data
8225 BEEP .1,25: RETURN
```

```
8230 REM
8235 REM     SUB N LOW-HIGHT
8240 REM
8245 LET d=256: GO TO 8255
8250 LET d=16
8255 LET h=INT (n/d)
8260 LET l=n-h*d: RETURN
8265 REM
8270 REM     SUB     POKE "**"
8275 REM
8280 POKE p,34: LET p=p+1
8285 POKE p,42: LET p=p+1
8290 POKE p,42: LET p=p+1
8295 POKE p,34: LET p=p+2
8300 LET ok=0: GO TO 8185
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 31999: LET a=32000
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM SUB DATA  LINE  FORMAT
9095 REM
9100 DATA "21","0F","27","CD"
9105 DATA "6E","19","28","41"
9110 DATA "1A","67","13","1A"
9115 DATA "6F","01","05","00"
9120 DATA "09","EB","21","0F"
9125 DATA "27","A7","ED","52"
9130 DATA "38","2F","21","00"
9135 DATA "5B","72","23","73"
9140 DATA "2B","D5","EB","ED"
9145 DATA "4B","02","5B","03"
9150 DATA "03","03","03","C5"
```

```
9155 DATA "D5","2A","4B","5C"
9160 DATA "1A","CD","08","0F"
9165 DATA "ED","53","4B","5C"
9170 DATA "D1","13","C1","0B"
9175 DATA "79","B0","20","EB"
9180 DATA "3E","02","CD","01"
9185 DATA "16","E1","C3","2D"
9190 DATA "18","CF","0F","**"
```

ASS.HEX-DEC DATA FORMAT

Questa routine è usata dal programma HEX-DEC DATA FORMAT. Il suo compito è quello di accodare al listato di questo programma Basic una nuova linea i cui dati sono stati depositati nel buffer di stampante.

LINEE DA 150 a 170:

Si cerca la linea Basic col numero 9999 (che è il massimo consentito) e se esiste si da un messaggio di errore.

LINEE DA 190 a 310:

Si carica nei registri HL il numero dell'ultima linea Basic e le si somma l'incremento (fissato in 5, ma è possibile modificarlo) e si controlla che il numero così ottenuto non superi 9999.

LINEE da 330 a 390:

La zona di RAM del buffer di stampante (da 5B00 a 5BFF esadecimali) viene utilizzata per memorizzare la nuova linea di Basic che andrà posta in coda al programma. Per cui si memorizza il nuovo numero di linea e,

LINEE da 410 a 580:

Prelevata la lunghezza della nuova linea Basic, si da inizio al trasferimento in fondo al programma aggiornando il puntatore di inizio variabili Basic.

LINEE da 590 a 620:

Si lista la nuova linea.

LINEE da 630 a 640:

Messaggio di errore.

```
7D00          0000          ORG   32000
              0010 ;
              0020 ;              "Routine"
              0030 ;              "Sistema"
              0040 ;
0F88          0050 INST     EQU   0F88H
196E          0060 FIND     EQU   196EH
1601          0070 OPEN     EQU   1601H
182D          0080 LIST     EQU   182DH
              0090 ;
              0100 ;              "Variab."
              0110 ;
5B00          0120 RAM      EQU   5B00H
5C4B          0130 VAR      EQU   5C4BH
              0140 ;
7D00 210F27   0150 INIZ     LD    HL,9999
7D03 CD6E19   0160          CALL  FIND
7D06 2841     0170          JR    Z,ERR
              0180 ;
7D08 1A       0190          LD    A,(DE)
7D09 67       0200          LD    H,A
7D0A 13       0210          INC   DE
7D0B 1A       0220          LD    A,(DE)
7D0C 6F       0230          LD    L,A
              0240 ;
7D0D 010500   0250          LD    BC,5
7D10 09       0260          ADD   HL,BC
7D11 EB       0270          EX    DE,HL
7D12 210F27   0280          LD    HL,9999
7D15 A7       0290          AND   A
7D16 ED52     0300          SBC   HL,DE
7D18 382F     0310          JR    C,ERR
              0320 ;
```

```
7D1A 21005B   0330           LD    HL,RAM
7D1D 72       0340           LD    (HL),D
7D1E 23       0350           INC   HL
7D1F 73       0360           LD    (HL),E
7D20 2B       0370           DEC   HL
7D21 D5       0380           PUSH  DE
7D22 EB       0390           EX    DE,HL
              0400 ;
7D23 ED4B025B 0410           LD    BC,(RAM+2)
7D27 03       0420           INC   BC
7D28 03       0430           INC   BC
7D29 03       0440           INC   BC
7D2A 03       0450           INC   BC
7D2B C5       0460 LB0       PUSH  BC
7D2C D5       0470           PUSH  DE
7D2D 2A4B5C   0480           LD    HL,(VAR)
7D30 1A       0490           LD    A,(DE)
7D31 CD880F   0500           CALL  INST
7D34 ED534B5C 0510           LD    (VAR),DE
7D38 D1       0520           POP   DE
7D39 13       0530           INC   DE
7D3A C1       0540           POP   BC
7D3B 0B       0550           DEC   BC
7D3C 79       0560           LD    A,C
7D3D B0       0570           OR    B
7D3E 20EB     0580           JR    NZ,LB0
7D40 3E02     0590           LD    A,2
7D42 CD0116   0600           CALL  OPEN
7D45 E1       0610           POP   HL
7D46 C32D18   0620           JP    LIST
7D49 CF       0630 ERR       RST   8
7D4A 0F       0640           DEFB  0FH
              0650           END
ERR     7D49
LB0     7D2B
INIZ    7D00
VAR     5C4B
RAM     5B00
LIST    182D
OPEN    1601
FIND    196E
INST    0F88
*       620B
```

HEX

Questo programma serve per fare il «DUMP» di zone di memoria cioè mostrarne sul video il contenuto eventualmente associato al relativo carattere ASCII.

Qualcuno obietterà che sia inutile complicare le cose ricorrendo al L/M. Ma non è così poichè analizzando il listato assembler si possono imparare numerose cose ad esempio:

- come predisporre il video alla scrittura (il che significa come aprire il canale logico associato al video),
- come passare variabili numeriche al L/M senza ricorrere a macchinosi POKE,
- e infine come realizzare in L/M la conversione di un valore numerico nella corrispondente rappresentazione ASCII esadecimale.

A tutto questo si deve associare la velocità propria del linguaggio macchina e allora certe soluzioni diventano più accettabili.

```
1000 REM      HEX-CHR$-DUMP
1005 REM    ESEMPIO  UTILIZZO
1010 REM ROUTINE DI  HEX-PRINT
1015 REM
1020 BORDER 1: PAPER 1: INK 7
1025 CLEAR 31999: GO SUB 1175
1030 FOR i=0 TO 7
1035 POKE USR "a"+i,0: NEXT i
1040 CLS : POKE USR "a"+4,16
1045 INPUT "INDIRIZZO INIZIALE ? ";i
1050 INPUT "INDIRIZZO FINALE ? ";f
1055 CLS : LET HX=32000
1060 LET s=5: LET SCRL=HX+57
1065 PRINT AT 0,9;"HEX-CHR$-DUMP"
1070 FOR j=i TO f STEP s
1075 PRINT AT 21,0;
1080 IF j<256 THEN  RANDOMIZE 0+USR HX
1085 RANDOMIZE j+USR HX
1090 PRINT " ";
```

```
1095 FOR k=0 TO s-1
1100 RANDOMIZE PEEK (j+k)+USR HX
1105 PRINT " ";: NEXT k
1110 PRINT "  ";
1115 FOR k=0 TO s-1
1120 LET p=PEEK (j+k)
1125 IF p>31 AND p<128 THEN  PRINT CHR$
p;: GO TO 1135
1130 PRINT CHR$ 144;
1135 PRINT " ";: NEXT k
1140 BEEP .01,60
1145 RANDOMIZE USR SCRL
1150 NEXT j
1155 GO TO 1045
1160 REM
1165 REM  HEX-CODE Caricatore
1170 REM
1175 LET a=32000
1180 PRINT AT 10,11;"ATTENDERE"
1185 BEEP .01,65: READ a$
1190 IF a$="**" THEN  RETURN
1195 LET h=CODE a$(1)-48
1200 LET h=h-7*(a$(1)>"@")
1205 LET l=CODE a$(2)-48
1210 LET l=l-7*(a$(2)>"@")
1215 POKE a,16*h+l: LET a=a+1
1220 GO TO 1185
1225 REM
1230 REM  DATI PROGRAMMA L.M.
1235 REM
1240 REM          HEX
1245 REM
1250 DATA "2A","65","5C","ED"
1255 DATA "5B","63","5C","A7"
1260 DATA "ED","52","7D","FE"
1265 DATA "05","DA","8B","28"
1270 DATA "3E","02","CD","01"
1275 DATA "16","CD","1B","7D"
1280 DATA "C3","28","2D","CD"
1285 DATA "99","1E","78","A7"
1290 DATA "C4","24","7D","79"
1295 DATA "F5","1F","1F","1F"
1300 DATA "1F","CD","2D","7D"
```

```
1305 DATA "F1","E6","0F","C6"
1310 DATA "30","FE","3A","38"
1315 DATA "02","C6","07","D7"
1320 DATA "C9","06","15","C3"
1325 DATA "00","0E","**"
```

```
                HEX-CHR$-DUMP

0096  52 4E C4 49 4E    R N . I N
009B  4B 45 59 A4 50    K E Y . P
00A0  C9 46 CE 50 4F    . F . P O
00A5  49 4E D4 53 43    I N . S C
00AA  52 45 45 4E A4    R E E N .
00AF  41 54 54 D2 41    A T T . A
00B4  D4 54 41 C2 56    . T A . V
00B9  41 4C A4 43 4F    A L . C O
00BE  44 C5 56 41 CC    D . V A .
00C3  4C 45 CE 53 49    L E . S I
00C8  CE 43 4F D3 54    . C O . T
00CD  41 CE 41 53 CE    A . A S .
00D2  41 43 D3 41 54    A C . A T
00D7  CE 4C CE 45 58    . L . E X
00DC  D0 49 4E D4 53    . I N . S
00E1  51 D2 53 47 CE    Q . S G .
00E6  41 42 D3 50 45    A B . P E
00EB  45 CB 49 CE 55    E . I . U
00F0  53 D2 53 54 52    S . S T R
00F5  A4 43 48 52 A4
```

ASS.HEX

Questa subroutine serve per stampare il contenuto del registro BC in esadecimale sullo schermo.

Veniamo al commento del listato:

LINEE da 160 a 220:

In queste linee di programma ci si assicura che il valore da stampare in Hex sia stato correttamente passato al linguaggio macchina. Ciò è stato fatto eseguendo un test sul contenuto della zona RAM definita dalle variabili di sistema STED e STBT (chiamate più o meno con lo stesso nome sul manuale Basic). Questa zona di RAM è utilizzata dal Basic per memorizzare il valore temporaneo di alcune variabili correntemente utilizzate. Dato che in ge-

nere abbiamo a che fare con variabili in floating-point ne risulta che, di ciascuna di esse, se ne da rappresentazione a 5 byte ed è questo il valore da controllare essendo una sola la variabile che passa dal Basic al L/M con l'istruzione del tipo RANDOMIZE var. + USR 32000.

Si dà un messaggio di errore nel caso la variabile non sia stata passata.

LINEE da 230 a 240:

In queste linee si apre il canale logico dedicato allo schermo : A=2. Se il registro A avesse contenuto ad esempio il numero 3 l'output sarebbe stato diretto alla stampante, con A = 0 oppure 1 si predisporrebbe l'uscita sulla parte bassa dello schermo.

LINEE da 250 a 260:

Qui è molto semplice: si chiama la routine di conversione e prima del ritorno al Basic si ripristina quella famosa zona di RAM, di cui prima abbiamo parlato, con un valore a caso.

LINEE da 270 a 430:

Si esegue il prelievo del numero che si vuole convertire e lo si passa alla routine di conversione.

LINEE da 440 a 450:

Di volta in volta si stampa sul canale logico selezionato in precedenza il carattere ASCII relativo al nostro numero in esadecimale.

```
7D00        0000        ORG   32000
            0010 ;
            0020 ;            "Routine"
            0030 ;            "Sistema"
            0040 ;
1E99        0050 LDBC   EQU   1E99H
2D28        0060 STACK  EQU   2D28H
288B        0070 ERR    EQU   288BH
1601        0080 OPEN   EQU   1601H
0010        0090 OUTC   EQU   10H
            0100 ;
            0110 ;            "Variab."
            0120 ;
```

```
5C65              0130 STED   EQU   5C65H
5C63              0140 STBT   EQU   5C63H
                  0150 ;
7D00 2A655C       0160 INIT   LD    HL,(STED)
7D03 ED5B635C     0170        LD    DE,(STBT)
7D07 A7           0180        AND   A
7D08 ED52         0190        SBC   HL,DE
7D0A 7D           0200        LD    A,L
7D0B FE05         0210        CP    5
7D0D DA8B28       0220        JP    C,ERR
7D10 3E02         0230        LD    A,2
7D12 CD0116       0240        CALL  OPEN
7D15 CD1B7D       0250        CALL  LDRG
7D18 C3282D       0260        JP    STACK
7D1B CD991E       0270 LDRG   CALL  LDBC
7D1E 78           0280        LD    A,B
7D1F A7           0290        AND   A
7D20 C4247D       0300        CALL  NZ,CONV
7D23 79           0310        LD    A,C
7D24 F5           0320 CONV   PUSH  AF
7D25 1F           0330        RRA
7D26 1F           0340        RRA
7D27 1F           0350        RRA
7D28 1F           0360        RRA
7D29 CD2D7D       0370        CALL  NIBLE
7D2C F1           0380        POP   AF
7D2D E60F         0390 NIBLE  AND   0FH
7D2F C630         0400        ADD   30H
7D31 FE3A         0410        CP    3AH
7D33 3802         0420        JR    C,OUT1
7D35 C607         0430        ADD   7
7D37 D7           0440 OUT1   RST   OUTC
7D38 C9           0450        RET
                  0460        END
OUT1    7D37
NIBLE   7D2D
CONV    7D24
LDRG    7D1B
INIT    7D00
STBT    5C63
STED    5C65
OUTC    0010
OPEN    1601
```

```
ERR     288B
STACK   2D28
LDBC    1E99
#       6062
```

LIST

Questo programma permette di listare un gruppo di linee di un programma Basic comprese tra due numeri di linea.

È noto infatti che lo Spectrum non permette di listare su schermo o su stampante un certo numero di linee. A volte però si sente questa necessità e qui viene dato un esempio di come si può aggirare l'ostacolo.

La solita routine in linguaggio macchina va memorizzata in RAM con un RUN 9000. A questo punto si può cancellare il programma Basic del caricatore del L/M e caricare quello che bisogna far stampare.

Il comando da dare è del tipo:

RANDOMIZE 100 + 255 * USR 32000

dove 100 e 255 sono due numeri di linea, iniziale e finale, tra le linee Basic da stampare, mentre 32000 è il solito indirizzo in RAM dove abbiamo allocato la routine in L/M.

Date un'occhiata anche a questa da cui si ricava, fra le varie cose, l'indirizzo in ROM ove il computer accede per listare una linea Basic.

```
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 31999: LET a=32000
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
```

```
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM        SUB LIST
9095 REM
9100 DATA "3E","03","18","02"
9105 DATA "3E","02","F5","2A"
9110 DATA "65","5C","ED","5B"
9115 DATA "63","5C","A7","ED"
9120 DATA "52","7D","FE","0A"
9125 DATA "DA","8B","28","F1"
9130 DATA "CD","01","16","21"
9135 DATA "00","00","22","5F"
9140 DATA "5C","CD","99","1E"
9145 DATA "ED","43","02","5B"
9150 DATA "CD","99","1E","C5"
9155 DATA "E1","CD","6E","19"
9160 DATA "7E","E6","C0","20"
9165 DATA "24","CD","9A","16"
9170 DATA "ED","53","00","5B"
9175 DATA "2B","ED","4B","02"
9180 DATA "5B","03","CD","80"
9185 DATA "19","30","12","16"
9190 DATA "20","D9","E5","D9"
9195 DATA "CD","6E","18","D7"
9200 DATA "D9","E1","D9","2A"
9205 DATA "00","5B","23","18"
9210 DATA "D4","CD","2B","2D"
9215 DATA "C3","2B","2D","**"
```

ASS.LIST

La routine seguente è utilizzata dal programma LIST, e permette di listare su video o su stampante un certo numero di linee. Il comando da dare è il seguente:

RANDOMIZE X + Y * USR RAM

dove: X = linea iniziale
Y = linea finale
RAM = indirizzo della routine in RAM

Commento:

LINEE da 230 a 420:

A seconda del punto di ingresso di questa routine, la stampa verrà fatta sul video o sulla stampante. Quindi:

PTER = stampante
PRINT = video

TEST per vedere se i numeri delle due linee sono stati correttamente specificati (altrimenti messaggio di errore).

Quindi si prelevano tali numeri dopo aver azzerato la variabile di sistema XPTR.

Questa normalmente contiene un indirizzo e se ne serve il computer in fase di editing. In caso di errore si memorizza in XPTR il punto in cui è avvenuto l'errore che verrà visualizzato con il simbolo '?' in reverse.

LINEE da 430 a 530:

Si ricerca la linea il cui numero è contenuto in HL e si fa un TEST per vedere se è l'ultima linea da stampare.

LINEE da 540 a 640:

Si stampa la linea : il registro D contiene il codice del carattere che si desidera stampare dopo il numero di linea. Normalmente è il carattere ' ' (spazio), ma il computer a seconda dei casi adopera '>'.

Prima di ritornare al Basic si devono ripristinare le zone di memoria svuotate dei numeri di linea suddetti. Ciò è necessario per non mandare in tilt il sistema.

```
7D00            0000        ORG  32000
                0010  ;
                0020  ;           "Routine"
```

```
                0030 ;              "Sistema"
                0040 ;
196E            0050 FIND   EQU     196EH
169A            0060 LINU   EQU     169AH
1980            0070 CPLN   EQU     1980H
186E            0080 PRIL   EQU     186EH
1601            0090 OPEN   EQU     1601H
1E99            0100 LDBC   EQU     1E99H
2D2B            0110 STBC   EQU     2D2BH
288B            0120 ERRM   EQU     288BH
                0130 ;
                0140 ;              "Variab."
                0150 ;
5B00            0160 LIN1   EQU     5B00H
5B02            0170 LIN2   EQU     5B02H
5C5F            0180 XPTR   EQU     5C5FH
5C65            0190 ENDS   EQU     5C65H
5C63            0200 BOTS   EQU     5C63H
0020            0210 CHAR   EQU     " "
                0220 ;
7D00 3E03       0230 PTER   LD      A,3
7D02 1802       0240        JR      TEST
7D04 3E02       0250 PRINT  LD      A,2
7D06 F5         0260 TEST   PUSH    AF
7D07 2A655C     0270        LD      HL,(ENDS)
7D0A ED5B635C   0280        LD      DE,(BOTS)
7D0E A7         0290        AND     A
7D0F ED52       0300        SBC     HL,DE
7D11 7D         0310        LD      A,L
7D12 FE0A       0320        CP      10
7D14 DA8B28     0330        JP      C,ERRM
7D17 F1         0340        POP     AF
7D18 CD0116     0350        CALL    OPEN
7D1B 210000     0360        LD      HL,0
7D1E 225F5C     0370        LD      (XPTR),HL
7D21 CD991E     0380        CALL    LDBC
7D24 ED43025B   0390        LD      (LIN2),BC
7D28 CD991E     0400        CALL    LDBC
7D2B C5         0410        PUSH    BC
7D2C E1         0420        POP     HL
7D2D CD6E19     0430 NEXT   CALL    FIND
7D30 7E         0440        LD      A,(HL)
7D31 E6C0       0450        AND     0C0H
```

```
7D33 2024     0460         JR    NZ,END
7D35 CD9A16   0470         CALL  LINU
7D38 ED53005B 0480         LD    (LIN1),DE
7D3C 2B       0490         DEC   HL
7D3D ED4B025B 0500         LD    BC,(LIN2)
7D41 03       0510         INC   BC
7D42 CD8019   0520         CALL  CPLN
7D45 3012     0530         JR    NC,END
7D47 1620     0540         LD    D,CHAR
7D49 D9       0550         EXX
7D4A E5       0560         PUSH  HL
7D4B D9       0570         EXX
7D4C CD6E18   0580         CALL  PRIL
7D4F D7       0590         RST   10H
7D50 D9       0600         EXX
7D51 E1       0610         POP   HL
7D52 D9       0620         EXX
7D53 2A005B   0630         LD    HL,(LIN1)
7D56 23       0640         INC   HL
7D57 18D4     0650         JR    NEXT
7D59 CD2B2D   0660 END     CALL  STBC
7D5C C32B2D   0670         JP    STBC
              0680         END
END     7D59
NEXT    7D2D
TEST    7D06
PRINT   7D04
PTER    7D00
CHAR    0020
BOTS    5C63
ENDS    5C65
XPTR    5C5F
LIN2    5B02
LIN1    5B00
ERRM    288B
STBC    2D2B
LDBC    1E99
OPEN    1601
PRIL    186E
CPLN    1980
LINU    169A
FIND    196E
#       6062
```

ERASE LINE

Il programma ERASE LINE serve a cancellare un blocco di linee Basic. Quello che si vede in figura è il listato del programma caricatore del L/M coi relativi dati. Dopo il RUN il programma metterà in memoria questa piccola routine a partire dall'indirizzo 32000.

Per vedere se tutto funziona, digitare il seguente comando:

RANDOMIZE 9000+9160*USR 32000

Il risultato sarà quello di cancellare le linee tra 9000 e 9160 comprese. Nel dare il comando è necessario interporre tra i due numeri di linea il segno "meno" oppure il "più" e poi il segno "per" o "diviso", insomma con priorità aritmetica più alta rispetto al precedente. Non così facendo si otterrà un messaggio di errore del tipo:

Q Parameter Error, 0:1

Se tutto funziona, si otterrà il seguente messaggio:

N Statement Lost, 0:1

Date un'occhiata al listato assembler per vedere come tutto ciò avviene.

```
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 31999: LET a=32000
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM SUB ERASE PROG. LINE
```

```
9095 REM
9100 DATA "2A","65","5C","ED"
9105 DATA "5B","63","5C","A7"
9110 DATA "ED","52","7D","FE"
9115 DATA "0A","DA","8B","28"
9120 DATA "CD","99","1E","03"
9125 DATA "C5","CD","99","1E"
9130 DATA "C5","E1","CD","6E"
9135 DATA "19","EB","E1","D5"
9140 DATA "CD","6E","19","D1"
9145 DATA "A7","ED","52","28"
9150 DATA "08","38","E2","EB"
9155 DATA "D5","C1","CD","E8"
9160 DATA "19","CF","16","**"
```

ASS. ERASE LINE

Il commento a questa routine è molto semplice. I codici sono stati memorizzati a partire dalla locazione 32000, ma come i più esperti avranno notato, la routine risulta essere rilocabile in qualunque zona della memoria.

Veniamo al commento:

LINEE da 150 a 210:

Test. Ci si assicura che il calcolatore abbia messo in memoria i due numeri di linea, altrimenti si da un messaggio di errore.

LINEE da 220 a 370:

Si prelevano questi numeri, si cercano gli indirizzi delle rispettive linee, si calcola il numero complessivo di byte occupati dalle linee comprese fra quelle indicate incluse e

LINEE da 380 a 430:

si cancellano producendo il messaggio N Statement Lost....

```
7D00            0000          ORG   32000
                0010 ;
                0020 ;              "Routine"
                0030 ;              "Sistema"
                0040 ;
288B            0050 ERRM     EQU   288BH
1E99            0060 LDBC     EQU   1E99H
196E            0070 FIND     EQU   196EH
19E8            0080 DELB     EQU   19E8H
                0090 ;
                0100 ;              "Variab."
                0110 ;
5C65            0120 ENDS     EQU   5C65H
5C63            0130 BOTS     EQU   5C63H
                0140 ;
7D00 2A655C     0150 INIZ     LD    HL,(ENDS)
7D03 ED5B635C   0160          LD    DE,(BOTS)
7D07 A7         0170          AND   A
7D08 ED52       0180          SBC   HL,DE
7D0A 7D         0190          LD    A,L
7D0B FE0A       0200          CP    10
7D0D DA8B28     0210 MIST     JP    C,ERRM
7D10 CD991E     0220          CALL  LDBC
7D13 03         0230          INC   BC
7D14 C5         0240          PUSH  BC
7D15 CD991E     0250          CALL  LDBC
7D18 C5         0260          PUSH  BC
7D19 E1         0270          POP   HL
7D1A CD6E19     0280          CALL  FIND
7D1D EB         0290          EX    DE,HL
7D1E E1         0300          POP   HL
7D1F D5         0310          PUSH  DE
7D20 CD6E19     0320          CALL  FIND
7D23 D1         0330          POP   DE
7D24 A7         0340          AND   A
7D25 ED52       0350          SBC   HL,DE
7D27 2808       0360          JR    Z,EXIT
7D29 38E2       0370          JR    C,MIST
7D2B EB         0380          EX    DE,HL
7D2C D5         0390          PUSH  DE
7D2D C1         0400          POP   BC
7D2E CDE819     0410          CALL  DELB
7D31 CF         0420 EXIT     RST   08H
```

```
7D32 16          0430      DEFB 16H
                 0440      END
EXIT    7D31
MIST    7D0D
INIZ    7D00
BOTS    5C63
ENDS    5C65
DELB    19E8
FIND    196E
LDBC    1E99
ERRM    288B
#       6030
```

RENUMBER

Il RENUMBER serve a rinumerare le linee di un programma come noi desideriamo in modo da ordinarle a partire da un numero voluto secondo un incremento stabilito.

È inclusa la possibilità di rinumerare le linee a partire dalla prima fino ad una linea all'interno del programma.

Il renumber qui proposto è scritto in Basic e si appoggia su due routine in L/M e sebbene scritto in Basic, esso tiene conto di tutti i GOTO, GOSUB, RESTORE, RUN, LIST, LLIST, LINE, seguiti da un numero di linea.

Il programma si divide in tre parti:

– la prima esegue un test in base ai parametri specificati nelle prime due linee (9950-9951) e cioè:

init = nuovo numero di linea dal quale partire a rinumerare,

step = passo tra due linee,

LF = linea alla quale si deve fermare la rinumerazione.

Infatti l'ultima linea rinumerata non deve avere numero superiore a quello della linea successiva e comunque non superiore o uguale a 9950, punto da cui inizia il nostro programma di rinumerazione.

Questo risiede in memoria insieme a quello da rinumerare quindi bisogna fare attenzione che quest'ultimo non abbia linee che interferiscano con il RENUMBER.

In tal caso bisognerà rinumerare manualmente queste poche linee.

Per come lavora questo RENUMBER, è necessario che i comandi GOTO, GOSUB, ecc.., siano seguiti da numeri di linea a 4 digit (esempio: 0001, 0102, ecc...), e non da espressioni numeriche di vario genere (esempio: GOTO a ∗ 10, GOSUB 1000 + a ∗ x).

Per testare tutto il programma, cercando ogni comando seguito da numero di linea, si fa uso di una routine in linguaggio macchina. Questo programmino intercetta tutti questi comandi controllandone la «sintassi» e se qualcosa va storto viene listata la linea con il cursore posto nel punto in cui bisogna apportare la correzione.

La seconda fase utilizza ancora la routine in L/M precedente, questa volta però essa fornisce l'indirizzo in RAM del comando intercettato che viene memorizzato nella variabile R$ insieme al suo numero di linea.

La terza fase è la rinumerazione vera e propria: qui il RENUMBER inizia a controllare i vecchi numeri di linea e in base alle informazioni contenute in R$, verifica se ad essi punta uno dei comandi (GOSUB, GOTO, RUN, ecc.) già visti. Se così è, si aggiorna la destinazione e comunque si rinumera la linea.

Il RENUMBER nella prima parte memorizza le due routine in L/M, di cui una si chiama FIND TOKEN e abbiamo già visto a che cosa serve, l'altra serve a cancellare le linee ed è qui introdotta per comodità.

Infatti dopo il RUN in memoria rimane solo il RENUMBER, mentre tutti i dati in L/M sono cancellati automaticamente.

A questo punto si può fare il MERGE del programma da rinumerare e dare il comando RUN 9950. Avvenuta la rinumerazione si può lasciare in memoria il programma rinumerato eseguendo il comando GOTO 9999.

Il RENUMBER in Basic, nonostante il L/M impiegato, è lento rispetto ad un equivalente programma totalmente in L/M.

Comunque può servire ai più smaliziati i quali, basandosi su ciò che qui è stato proposto, potrebbero trovare lo spunto per costruirsi un completo RENUMBER in L/M.

```
9000 REM
9005 REM  HEX-CODE Caricatore
```

```
9010 REM
9030 CLEAR 31999: GO SUB 9992: LET a=Nex
t
9035 READ a$
9040 IF a$="**" THEN  RANDOMIZE 0+9949*U
SR (Next+191)
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM SUB  FIND TOKEN
9095 REM
9100 DATA "2A","00","5B","18","14","2A"
9105 DATA "53","5C","01","DE","26","CD"
9110 DATA "80","19","D0","CD","9A","16"
9115 DATA "ED","53","02","5B","CD","BC"
9120 DATA "18","7E","CD","B6","18","FE"
9125 DATA "EA","20","09","2A","02","5B"
9130 DATA "23","CD","6E","19","18","DE"
9135 DATA "FE","EC","28","1F","FE","ED"
9140 DATA "28","1B","FE","E5","28","17"
9145 DATA "FE","F7","28","13","FE","CA"
9150 DATA "28","0F","FE","F0","28","0B"
9155 DATA "FE","E1","28","07","23","FE"
9160 DATA "0D","28","BD","18","CC","23"
9165 DATA "7E","FE","0D","28","F3","2B"
9170 DATA "FE","3A","38","0E","7E","FE"
9175 DATA "CA","28","E9","23","22","00"
9180 DATA "5B","E5","18","31","18","9A"
9185 DATA "E5","CD","BA","18","CD","B6"
9190 DATA "18","22","00","5B","20","23"
9195 DATA "FE","3A","28","04","FE","0D"
9200 DATA "20","1B","2B","2B","7E","2B"
9205 DATA "6E","67","CD","6E","19","20"
9210 DATA "10","01","DE","26","CD","80"
9215 DATA "19","30","08","C1","3A","04"
9220 DATA "5B","A7","20","D0","C9","3E"
9225 DATA "02","32","04","5B","CD","01"
```

```
9230 DATA "16","E1","D9","E5","D9","23"
9235 DATA "22","5F","5C","2A","02","5B"
9240 DATA "CD","6E","19","16","23","CD"
9245 DATA "6E","18","D7","D9","E1","D9"
9250 DATA "11","10","00","21","90","1A"
9255 DATA "CD","B5","03","18","A5"
9260 REM
9265 REM SUB   ERASE LINE
9270 REM
9275 DATA "2A","65","5C","ED","5B","63"
9280 DATA "5C","A7","ED","52","7D","FE"
9285 DATA "0A","DA","8B","28","CD","99"
9290 DATA "1E","03","C5","CD","99","1E"
9295 DATA "C5","E1","CD","6E","19","EB"
9300 DATA "E1","D5","CD","6E","19","D1"
9305 DATA "A7","ED","52","28","08","38"
9310 DATA "E2","EB","D5","C1","CD","E8"
9315 DATA "19","CF","16","**"
9320 REM
9325 REM          RENUMBER
9330 REM
9950 LET init=1000: LET step=5
9951 LET lf=9950: LET x=23635
9952 LET nl=9950: LET Err=23300
9953 LET c=256: POKE Err,1
9954 GO SUB 9992: LET z$="0000"
9955 GO SUB 9987: RANDOMIZE USR Run
9956 IF PEEK Err=2 THEN  STOP
9957 POKE Err,0: REM Reset Err
9958 DEF FN a(x)=c*PEEK (x+1)+PEEK x
9959 DEF FN b(y)=c*PEEK y+PEEK (y+1)
9960 GO SUB 9993: GO SUB 9988
9961 REM
9962 REM
9963 LET r$="": LET i=USR Run
9964 IF i=nl THEN  GO TO 9970
9965 LET r$=r$+STR$ i
9966 FOR l=i+1 TO i+4
9967 LET r$=r$+CHR$ PEEK l
9968 NEXT l
9969 LET i=USR Next: GO TO 9964
9970 GO SUB 9989
9971 LET y=FN a(x)
```

```
9972 LET linea=FN b(y)
9973 IF linea>=lf THEN  BEEP .2,30: LIST
 : STOP
9974 LET lung=FN a(y+2)
9975 FOR i=1 TO LEN r$ STEP 9
9976 IF VAL r$(i+5 TO i+8)<>linea THEN
GO TO 9979
9977 IF linea<lf THEN  GO SUB 9983
9978 REM
9979 NEXT i: IF linea>=lf THEN  GO TO 99
82
9980 GO SUB 9991
9981 POKE y,hi: POKE y+1,low: LET init=i
nit+step
9982 LET y=y+lung+4: GO TO 9972
9983 LET a$=STR$ init: LET h=VAL r$(i TO
 i+4)
9984 LET z=4-LEN a$: IF z THEN  LET a$=z
$( TO z)+a$
9985 FOR j=1 TO 4: POKE h+j,CODE a$(j):
NEXT j
9986 GO SUB 9991: POKE h+8,low: POKE h+9
,hi: RETURN
9987 CLS : PRINT AT 11,13; FLASH 1;"Test
": RETURN
9988 PRINT AT 11,10; FLASH 1;"Find TOKEN
": RETURN
9989 PRINT AT 11,9; FLASH 1;"Rinumerazio
ne": RETURN
9990 REM
9991 LET hi=INT (init/c): LET low=init-c
*hi: RETURN
9992 LET Next=32000: LET Run=Next+5: RETURN
9993 LET y=FN a(x): LET h=init
9994 LET linea=FN b(y): LET y=y+FN a(y+2
)+4
9995 IF h>=lf OR h>=nl THEN  RANDOMIZE U
SR 10379
9996 IF linea>=lf OR linea>=nl THEN  RETURN
9997 LET h=h+step: GO TO 9994
9998 REM
9999 RANDOMIZE 9950+9999*USR (Next+191)
```

FIND TOKEN

Questa routine è usata dal programma Basic RENUMBER e serve a trovare tutti quei comandi che prevedono come argomento un numero di linea a 4 digit (esempio: GOSUB 0005, GOTO 1129, ecc..), non un numero calcolato (esempio: GOSUB 10*a), restituendo al programma Basic che la gestisce, il relativo indirizzo in RAM.

I comandi intercettati sono:

Comando	Codice	Comando	Codice
GOTO	EC	LINE	CA
GOSUB	ED	LIST	F0
RESTORE	E5	LLIST	E1
RUN	F7	REM	EA

REM è un caso anomalo: quando viene incontrato si salta ad esaminare la linea successiva.

Commento al listato assembler:

LINEE da 240 a 400:

Vi sono due punti di ingresso. Quello siglato INIT serve al momento di lanciare la routine, mentre quello siglato CONT serve per le elaborazioni successive.

All'indirizzo NEXT vi si accede quando si desidera analizzare una nuova linea Basic. In tal caso se ne confronta il numero con quello da noi stabilito (9950 in questa applicazione) ritornando al Basic se il confronto ha successo o se è superiore, altrimenti si carica il numero della linea in esame e lo si salva in una zona di RAM.

La fase successiva è quella di prelevare un carattere dalla linea confrontarlo con il codice EA (=REM) e se è così si passa alla linea successiva altrimenti si prosegue al confronto degli altri codici.

LINEE da 410 a 580:

Se uno di questi confronti ha successo, significa che è presente un comando tra quelli da cercare, altrimenti si passa al prossimo carattere e alla prossima linea.

LINEE DA 590 a 730:

Qui si esamina il carattere successivo al comando intercettato per vedere se esso è un digit (un carattere tra 0 e 9). Se così è si passa alla fase di TEST, altrimenti si salta alla routine di errore o alla prossima linea.

LINEE da 740 a 980:

Fase di TEST. Dopo il byte del comando ci deve essere un numero a 4 digit e nient'altro dopo di esso (esempio: potremmo avere a che fare con un GOTO 1000*a). Inoltre questo numero deve far riferimento ad una linea effettiva esistente.

È noto che lo Spectrum accetta comandi del tipo GOSUB 0120, anche se la linea 120 non esiste: l'elaborazione prosegue infatti con la linea successiva.

Per correttezza ho preferito che ogni riferimento lo si facesse a linee esistenti e con numero non superiore a 9950.

LINEE da 990 a 1200:

Routine di errore. Si stampa la linea da correggere emettendo un "beep" di avvertimento.

```
7D00        0000        ORG  32000
            0010 ;
            0020 ;           "Routine"
            0030 ;           "Sistema"
            0040 ;
18BC        0050 INC3  EQU   18BCH
18BA        0060 INC5  EQU   18BAH
18B6        0070 INCP  EQU   18B6H
196E        0080 FIND  EQU   196EH
169A        0090 LINU  EQU   169AH
1980        0100 CPLN  EQU   1980H
186E        0110 PRIL  EQU   186EH
1601        0120 OPEN  EQU   1601H
03B5        0130 BEEP  EQU   03B5H
            0140 ;
            0150 ;           "Variab."
            0160 ;
5B00        0170 ADDL  EQU   5B00H
5B02        0180 LINE  EQU   5B02H
5B04        0190 EROR  EQU   5B04H
```

```
5C53              0200 PROG  EQU   5C53H
5C5F              0210 XPTR  EQU   5C5FH
26DE              0220 MAXL  EQU   9950
                  0230 ;
7D00 2A005B       0240 CONT  LD    HL,(ADDL)
7D03 1814         0250       JR    CHAR
7D05 2A535C       0260 INIT  LD    HL,(PROG)
7D08 01DE26       0270 NEXT  LD    BC,MAXL
7D0B CD8019       0280       CALL  CPLN
7D0E D0           0290       RET   NC
7D0F CD9A16       0300       CALL  LINU
7D12 ED53025B     0310       LD    (LINE),DE
7D16 CDBC18       0320       CALL  INC3
7D19 7E           0330 CHAR  LD    A,(HL)
7D1A CDB618       0340       CALL  INCP
7D1D FEEA         0350       CP    0EAH
7D1F 2009         0360       JR    NZ,GO
7D21 2A025B       0370       LD    HL,(LINE)
7D24 23           0380       INC   HL
7D25 CD6E19       0390       CALL  FIND
7D28 18DE         0400       JR    NEXT
7D2A FEEC         0410 GO    CP    0ECH
7D2C 281F         0420       JR    Z,DIGI
7D2E FEED         0430       CP    0EDH
7D30 281B         0440       JR    Z,DIGI
7D32 FEE5         0450       CP    0E5H
7D34 2817         0460       JR    Z,DIGI
7D36 FEF7         0470       CP    0F7H
7D38 2813         0480       JR    Z,DIGI
7D3A FECA         0490       CP    0CAH
7D3C 280F         0500       JR    Z,DIGI
7D3E FEF0         0510       CP    0F0H
7D40 280B         0520       JR    Z,DIGI
7D42 FEE1         0530       CP    0E1H
7D44 2807         0540       JR    Z,DIGI
7D46 23           0550 INCR  INC   HL
7D47 FE0D         0560       CP    0DH
7D49 28BD         0570       JR    Z,NEXT
7D4B 18CC         0580       JR    CHAR
7D4D 23           0590 DIGI  INC   HL
7D4E 7E           0600       LD    A,(HL)
7D4F FE0D         0610       CP    0DH
7D51 28F3         0620       JR    Z,INCR
```

```
7D53 2B      0630      DEC  HL
7D54 FE3A    0640      CP   3AH
7D56 380E    0650      JR   C,TEST
7D58 7E      0660      LD   A,(HL)
7D59 FECA    0670      CP   0CAH
7D5B 28E9    0680      JR   Z,INCR
7D5D 23      0690      INC  HL
7D5E 22005B  0700      LD   (ADDL),HL
7D61 E5      0710      PUSH HL
7D62 1831    0720      JR   ERRR
7D64 189A    0730 GOTO JR   CONT
7D66 E5      0740 TEST PUSH HL
7D67 CDBA18  0750      CALL INC5
7D6A CDB618  0760      CALL INCP
7D6D 22005B  0770      LD   (ADDL),HL
7D70 2023    0780      JR   NZ,ERRR
7D72 FE3A    0790      CP   3AH
7D74 2804    0800      JR   Z,ON
7D76 FE0D    0810      CP   0DH
7D78 201B    0820      JR   NZ,ERRR
7D7A 2B      0830 ON   DEC  HL
7D7B 2B      0840      DEC  HL
7D7C 7E      0850      LD   A,(HL)
7D7D 2B      0860      DEC  HL
7D7E 6E      0870      LD   L,(HL)
7D7F 67      0880      LD   H,A
7D80 CD6E19  0890      CALL FIND
7D83 2010    0900      JR   NZ,ERRR
7D85 01DE26  0910      LD   BC,MAXL
7D88 CD8019  0920      CALL CPLN
7D8B 3008    0930      JR   NC,ERRR
7D8D C1      0940      POP  BC
7D8E 3A045B  0950      LD   A,(EROR)
7D91 A7      0960      AND  A
7D92 20D0    0970      JR   NZ,GOTO
7D94 C9      0980      RET
7D95 3E02    0990 ERRR LD   A,2
7D97 32045B  1000      LD   (EROR),A
7D9A CD0116  1010      CALL OPEN
7D9D E1      1020      POP  HL
7D9E D9      1030      EXX
7D9F E5      1040      PUSH HL
7DA0 D9      1050      EXX
```

```
7DA1 23        1060       INC  HL
7DA2 225F5C    1070       LD   (XPTR),HL
7DA5 2A025B    1080       LD   HL,(LINE)
7DA8 CD6E19    1090       CALL FIND
7DAB 1623      1100       LD   D,"#"
7DAD CD6E18    1110       CALL PRIL
7DB0 D7        1120       RST  10H
7DB1 D9        1130       EXX
7DB2 E1        1140       POP  HL
7DB3 D9        1150       EXX
7DB4 111000    1160 BELL  LD   DE,16
7DB7 21901A    1170       LD   HL,1A90H
7DBA CDB503    1180       CALL BEEP
7DBD 18A5      1190       JR   GOTO
               1200       END
BELL    7DB4
ERRR    7D95
ON      7D7A
TEST    7D66
GOTO    7D64
DIGI    7D4D
INCR    7D46
GO      7D2A
CHAR    7D19
NEXT    7D08
INIT    7D05
CONT    7D00
MAXL    26DE
XPTR    5C5F
PROG    5C53
EROR    5B04
LINE    5B02
ADDL    5B00
BEEP    03B5
OPEN    1601
PRIL    186E
CPLN    1980
LINU    169A
FIND    196E
INCP    18B6
INC5    18BA
INC3    18BC
#       6062
```

AUTONUMBER

Quello che segue è il listato del programma AUTONUMBER.

Serve per avere automaticamente il numero di linea sullo schermo quando si introduce un programma Basic.

Per utilizzare il programma basta eseguire questa istruzione:

RANDOMIZE USR 32341

Da questo momento in poi ogni volta che si preme il tasto RETURN, comparirà in basso il numero di linea successiva con incrementi di 10.

In ogni momento si può cambiare il numero della linea che ci viene stampata semplicemente cancellando tale numero e introducendo quello desiderato.

L'incremento è fissato in 10, ma è possibile modificarlo con un POKE 32507,i dove i=incremento (dal listato assembler si vede facilmente ove questa modifica può essere fatta: LINEA 550 ove si carica il registro DE con il numero 000A che è l'incremento uguale a 10).

Per togliere l'auto digitare il seguente comando:

RANDOMIZE USR 32334

Se il numero di linea eccede 9999, due punti ':' verranno stampati al posto del primo digit impedendo l'introduzione della linea.

La gestione dell'AUTONUMBER è stata messa sotto la routine d'interrupt.

Un'occhiata al listato assembler può dare un'idea di come ciò sia possibile.

```
1000 REM        COMANDI
1010 REM   RANDOMIZE USR 32341
1020 REM   PER INIZIARE AUTO
1030 REM
1040 REM   RANDOMIZE USR 32334
1050 REM  PER  ELIMINARE AUTO
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 32333: LET a=32334
```

```
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM     SUB AUTONUMBER
9095 REM
9100 DATA "3E","3F","ED","47"
9105 DATA "ED","56","C9","3E"
9110 DATA "28","ED","47","ED"
9115 DATA "5E","C9","FF","F3"
9120 DATA "F5","C5","D5","E5"
9125 DATA "3A","DD","7E","A7"
9130 DATA "20","24","3A","82"
9135 DATA "5C","FE","20","20"
9140 DATA "4F","3A","83","5C"
9145 DATA "FE","17","20","48"
9150 DATA "3A","08","5C","FE"
9155 DATA "0C","28","41","3A"
9160 DATA "04","5C","FE","0D"
9165 DATA "28","03","3C","20"
9170 DATA "37","3E","04","32"
9175 DATA "DD","7E","3A","DD"
9180 DATA "7E","3D","32","DD"
9185 REM
9190 REM      SECONDA PARTE
9195 REM
9200 DATA "7E","2A","49","5C"
9205 DATA "11","0A","00","19"
9210 DATA "01","18","FC","CD"
9215 DATA "C4","7E","FE","03"
9220 DATA "28","1A","01","9C"
9225 DATA "FF","CD","C4","7E"
9230 DATA "FE","02","28","10"
9235 DATA "01","F6","FF","CD"
9240 DATA "C4","7E","FE","01"
9245 DATA "28","06","01","FF"
```

```
9250 DATA "FF","CD","C4","7E"
9255 DATA "E1","D1","C1","F1"
9260 DATA "FB","C9","AF","09"
9265 DATA "3C","38","FC","ED"
9270 DATA "42","3D","C6","30"
9275 DATA "32","08","5C","3A"
9280 DATA "3B","5C","CB","EF"
9285 DATA "32","3B","5C","3A"
9290 DATA "DD","7E","C9","**"
```

ASS. AUTONUMBER

Per capire come funziona questa routine di AUTONUMBER occorre premettere qualche informazione su come si gestiscono gli interrupts mascherabili con lo Z-80.

Ogni volta che si richiede un interrupt (IRQ), lo Z-80 risponde in accordo con l'IM (interrupt mode) che gli è stato imposto.

Nel modo IM 1, lo Z-80 risponde all'IRQ eseguendo la routine posta in memoria (ROM oppure RAM) a partire dall'indirizzo esadecimale 0038H (56 in decimale).

I modi IM 0 e IM 2 sono strettamente collegati alla gestione degli IRQ da parte di dispositivi Hardware esterni e vengono detti interrupt vettorizzati.

Vettorizzati significa che in qualche modo bisogna costruirsi un indirizzo a 16 bit a cui fare riferimento.

Nel modo IM 2 si fa riferimento a delle tabelle il cui indirizzo in memoria è generato dalla combinazione di due dati:

Il primo è contenuto nel registro I (interrupt register), parte alta dell'indirizzo.

Il secondo è fornito dalla periferica stessa ed è la parte bassa dell'indirizzo.

A questo punto lo Z-80 carica nel PC (program counter) i due byte successivi posti al suddetto indirizzo e serve l'interrupt.

Nel modo IM 0 avviene qualcosa di simile con la differenza che la periferica pone sul bus dei dati un valore che in generale è uno dei possibili 8 codici di restart in pagina zero.

Di questi tre tipi lo Spectrum pare utilizzare quello che è chiamato modo 1 (IM 1).

Con un piccolo trucco è possibile far funzionare lo Spectrum nel modo IM 2.

Infatti la periferica che dovrebbe fornire il byte basso dell'indirizzo della tabella non esiste, per cui tale byte è sempre posto uguale a 0FFH (255 in decimale).

Basta a questo punto caricare il registro I con un valore, nel nostro caso 28H, per costringere il computer a cercare all'indirizzo 28FFH i dati da caricare nel program counter.

All'indirizzo 28FFH (che è in ROM) e successivo vi sono questi numeri 5C 7E che sono proprio l'indirizzo da cui parte la nostra routine di autonumber.

Commento:

LINEE da 210 a 240:

Sono due subroutine che predispongono i due modi di interrupt, rispettivamente IM 1 e IM 2.

LINEE da 260 a 480:

Si esegue la normale routine di interrupt e si inizia quella di autonumber dopo aver salvato i registri, controllato lo spazio nel buffer di editing e testato il tasto premuto che deve essere un carriage return (0DH) per dare inizio alla numerazione automatica.

LINEE da 490 a 760:

Il numero che vogliamo stampare è di 4 digit e con questo si inizializza la variabile RAM.
In base all'ultimo numero stampato (contenuto in LNUM) si calcola quello nuovo secondo l'incremento nel registro DE (linea 550).

Si stampa il numero richiamando 4 volte la subroutine ASCI, si ripristinano i registri e si esce.

LINEE DA 780 a 920:

Questa è la subroutine ASCI che in base ai valori contenuti nei registri HL e BC fornisce ogni volta l'esatto digit da stampare.
Questo una volta determinato occorre stamparlo e ciò viene fatto memorizzandolo nella variabile LKEY e simulando la pressione del relativo tasto settando il bit 5 della variabile FLAG.

```
7E4E          0000          ORG   7E4EH
              0010 ;
              0020 ;              "Routine"
              0030 ;              "Sistema"
              0040 ;
0038          0050 IRQ     EQU   0038H
              0060 ;
              0070 ;              "Variab."
              0080 ;
5C04          0090 KEYD    EQU   5C04H
5C08          0100 LKEY    EQU   5C08H
5C3B          0110 FLAG    EQU   5C3BH
5C49          0120 LNUM    EQU   5C49H
5C82          0130 COLN    EQU   5C82H
5C83          0140 LINE    EQU   5C83H
              0150 ;
7E4E 3E3F     0160 INT1    LD    A,3FH
7E50 ED47     0170         LD    I,A
7E52 ED56     0180         IM1
7E54 C9       0190         RET
              0200 ;
7E55 3E28     0210 INT2    LD    A,28H
7E57 ED47     0220         LD    I,A
7E59 ED5E     0230         IM2
7E5B C9       0240         RET
              0250 ;
7E5C FF       0260 INIT    RST   IRQ
7E5D F3       0270         DI
7E5E F5       0280         PUSH  AF
7E5F C5       0290         PUSH  BC
7E60 D5       0300         PUSH  DE
7E61 E5       0310         PUSH  HL
7E62 3ADD7E   0320         LD    A,(RAM)
7E65 A7       0330         AND   A
7E66 2024     0340         JR    NZ,LB0
7E68 3A825C   0350         LD    A,(COLN)
7E6B FE20     0360         CP    20H
7E6D 204F     0370         JR    NZ,OUT
7E6F 3A835C   0380         LD    A,(LINE)
7E72 FE17     0390         CP    17H
7E74 2048     0400         JR    NZ,OUT
7E76 3A085C   0410         LD    A,(LKEY)
7E79 FE0C     0420         CP    0CH
```

```
7E7B 2841      0430      JR   Z,OUT
7E7D 3A045C    0440      LD   A,(KEYD)
7E80 FE0D      0450      CP   0DH
7E82 2803      0460      JR   Z,LB1
7E84 3C        0470      INC  A
7E85 2037      0480      JR   NZ,OUT
7E87 3E04      0490 LB1  LD   A,04H
7E89 32DD7E    0500      LD   (RAM),A
7E8C 3ADD7E    0510 LB0  LD   A,(RAM)
7E8F 3D        0520      DEC  A
7E90 32DD7E    0530      LD   (RAM),A
7E93 2A495C    0540      LD   HL,(LNUM)
7E96 110A00    0550      LD   DE,0AH
7E99 19        0560      ADD  HL,DE
7E9A 0118FC    0570      LD   BC,64536
7E9D CDC47E    0580      CALL ASCI
7EA0 FE03      0590      CP   03H
7EA2 281A      0600      JR   Z,OUT
7EA4 019CFF    0610      LD   BC,65436
7EA7 CDC47E    0620      CALL ASCI
7EAA FE02      0630      CP   02H
7EAC 2810      0640      JR   Z,OUT
7EAE 01F6FF    0650      LD   BC,65526
7EB1 CDC47E    0660      CALL ASCI
7EB4 FE01      0670      CP   01H
7EB6 2806      0680      JR   Z,OUT
7EB8 01FFFF    0690      LD   BC,65535
7EBB CDC47E    0700      CALL ASCI
7EBE E1        0710 OUT  POP  HL
7EBF D1        0720      POP  DE
7EC0 C1        0730      POP  BC
7EC1 F1        0740      POP  AF
7EC2 FB        0750      EI
7EC3 C9        0760      RET
               0770 ;
7EC4 AF        0780 ASCI XOR  A
7EC5 09        0790 SOM  ADD  HL,BC
7EC6 3C        0800      INC  A
7EC7 38FC      0810      JR   C,SOM
7EC9 ED42      0820      SBC  HL,BC
7ECB 3D        0830      DEC  A
7ECC C630      0840      ADD  30H
7ECE 32085C    0850      LD   (LKEY),A
```

```
7ED1 3A3B5C     0860          LD    A,(FLAG)
7ED4 CBEF       0870          SET   5,A
7ED6 323B5C     0880          LD    (FLAG),A
7ED9 3ADD7E     0890          LD    A,(RAM)
7EDC C9         0900          RET
                0910 ;
7EDD 00         0920 RAM      DEFB 0
                0930          END

RAM     7EDD
SOM     7EC5
ASCI    7EC4
OUT     7EBE
LB0     7E8C
LB1     7E87
INIT    7E5C
INT2    7E55
INT1    7E4E
LINE    5C83
COLN    5C82
LNUM    5C49
FLAG    5C3B
LKEY    5C08
KEYD    5C04
IRQ     0038
#       6062
```

FILE SCANNER

FILE SCANNER è un programma che è in grado di esaminare un file registrato su nastro e stabilirne la natura, la lunghezza e in generale è in grado di dare informazioni aggiuntive proprio in base al tipo di file che si sta trattando.

Il cuore di tutto sta in quei pochi dati nelle due linee di DATA in fondo al programma. Nel loro insieme realizzano una piccola routine in L/M che di ogni file carica solo la prima parte, l'HEADER.

Questo contiene tutte le informazioni circa il tipo, cioè se si tratta di un programma BASIC, un file binario, dati di un array alfanumerico e così via.

Di un programma in BASIC è in grado di stabilire la lunghezza totale, la lunghezza occupata dalle variabili, l'eventuale linea di autorun.

Di un programma in L/M è possibile conoscere la locazione iniziale e sempre la sua lunghezza.

Di un array alfanumerico/numerico è in grado di dare la sua dimensione e il nome della variabile associata.

```
1000 BORDER 1: INK 7: PAPER 1
1005 LET c=23296: LET p=c+14
1010 LET q=19: CLS
1015 FOR i=c TO c+17
1020 POKE i,0: NEXT i
1025 FOR i=p TO p+11: READ d
1030 POKE i,d: NEXT i
1035 RANDOMIZE USR p
1040 LET t=PEEK c: LET d=PEEK (c+14)
1045 LET lu=256*PEEK (c+12)+PEEK (c+11)
1050 LET ln=256*PEEK (c+14)+PEEK (c+13)
1055 LET vr=256*PEEK (c+16)+PEEK (c+15)
1060 IF t=0 THEN  PRINT "Program";TAB 16
;":";TAB q;
1065 IF t=1 THEN  PRINT "Character Array
 :";TAB q;
1070 IF t=2 THEN  PRINT "Number Array
 :";TAB q;
1075 IF t=3 THEN  PRINT "Bytes";TAB q-3;
":";TAB q;
1080 IF PEEK (c+1)=255 THEN  GO TO 1095
1085 FOR i=c+1 TO c+10
1090 PRINT CHR$ PEEK i;: NEXT i
1095 PRINT
1100 IF t=0 THEN  PRINT "Linea  Autorun
 :";TAB q;: IF ln>=1e4 THEN  PRINT  FLAS
H 1;"NESSUNA": GO TO 1120
1105 IF t=0 THEN  PRINT ln
1110 IF t=3 THEN  PRINT "Locazione inizi
o: ";TAB q;ln
1115 IF t=2 OR t=1 THEN  PRINT "Variabil
e array : ";CHR$ (d-(192 AND t=2)-(128 A
ND t=1)+96)+("$" AND t=2)+"( )"
1120 PRINT "Lunghezza file  :";TAB q;lu;
TAB 25;"Bytes"
```

```
1125 IF t=0 THEN  PRINT "Lunghezza Var.
  :";TAB q;lu-vr;TAB 25;"Bytes"
1130 DATA 17,17,0,175,55,221
1135 DATA 33,0,91,195,86,05
```

DETECT 1.0

Questo programma, mi si permetta l'immodestia, è quello che fra tutti ritengo il più utile e meglio realizzato. Come sempre però, ciascuno secondo le proprie esigenze può apportare le modifiche necessarie basandosi sul listato assembler.

Ma veniamo al dunque. A cosa serve?

Serve a due cose:

1) a fare il backup dei programmi prelevandoli da una cassetta e registrandoli su un'altra tramite computer. Per dirla in parole povere serve per fare la copia dei programmi che con i metodi tradizionali non si riescono a duplicare.

Attenzione:

Come sempre le copie che si ottengono devono essere realizzate per proprio uso e non a fini poco leciti, quali la rivendita, e ciò per i noti motivi di copyright.

2) il secondo uso è quello di rivelare l'inizio e la lunghezza dei file binari, la lunghezza e la linea di autorun dei programmi Basic, e le caratteristiche dei file contenenti array alfanumerici.

Con tale programma scritto in linguaggio macchina si possono duplicare files di lunghezza massima di 40798 (9F5E H) bytes. È evidente che tale programma è utilizzabile su uno Spectrum 48k.

Tuttavia nelle pagine di questo libro è presentato un programma di pochi byte che può essere adattato ad uno Spectrum 16k con ovvie limitazioni di spazio.

Nell'insieme DETECT 1.0 si compone di due parti:

1) la prima è quella che metterà a dura prova la vostra pazienza. Si tratta cioè di digitare i 700 e più bytes del programma in L/M.
   Fatto questo si salva il tutto su nastro. Occorre fare molta attenzione nel digitare questa parte di programma poiché anche un piccolo errore può essere fatale ed è abbastanza oneroso controllare tutti i codici. Dato il RUN, il programma si ferma alla linea 9040, ciò significa che il codice macchina è stato memorizzato a partire dalla locazione 64798, e bisogna salvarlo con:

   SAVE "DETECT 1.0" CODE 64798,738

2) in un secondo tempo basteranno 3 righe di Basic, visibili nel primo listato, per richiamare in memoria il codice oggetto e lanciarlo;

3) l'uso del programma è poi molto semplice poiché esso chiederà di volta in volta le operazioni da compiere;

4) il lancio del programma avviene con un RANDOMIZE USR 64798, e da questo momento in poi il computer aspetta il file da duplicare.

```
10 INK 9: BORDER 7: PAPER 7
20 CLEAR 23999: LOAD ""CODE
30 RANDOMIZE USR 64798
```

```
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 64797: LET a=64798
9035 READ a$
9040 IF a$="**" THEN  STOP
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
```

```
9090 REM   DETECT 1.0 BACKUP
9095 REM
9100 DATA "CD","28","FE","3E","06","CD"
9105 DATA "47","FE","AF","37","DD","21"
9110 DATA "00","5B","11","11","00","CD"
9115 DATA "56","05","30","EA","3E","0A"
9120 DATA "CD","47","FE","11","01","5B"
9125 DATA "1A","FE","FF","28","06","01"
9130 DATA "0A","00","CD","3C","20","AF"
9135 DATA "CD","47","FE","3E","11","D7"
9140 DATA "3A","00","5B","F5","D7","F1"
9145 DATA "F5","11","A2","09","CD","0A"
9150 DATA "0C","3E","05","CD","47","FE"
9155 DATA "3E","01","CD","47","FE","ED"
9160 DATA "4B","0B","5B","3E","0A","CD"
9165 DATA "D1","FD","F1","FE","00","20"
9170 DATA "14","3E","02","CD","47","FE"
9175 DATA "ED","4B","0D","5B","CB","78"
9180 DATA "28","12","3E","04","C4","47"
9185 REM
9190 REM   DETECT SECONDA PARTE
9195 REM
9200 DATA "FE","18","10","FE","03","20"
9205 DATA "0C","CD","47","FE","ED","4B"
9210 DATA "0D","5B","3E","0D","CD","D1"
9215 DATA "FD","CD","39","FE","DD","21"
9220 DATA "C0","5D","3E","FF","37","CD"
9225 DATA "56","05","30","F1","3E","07"
9230 DATA "CD","FC","FD","3E","0B","CD"
9235 DATA "4C","FE","AF","CD","62","FE"
9240 DATA "3E","08","CD","FC","FD","FE"
9245 DATA "6C","CA","1E","FD","FE","73"
9250 DATA "28","E9","FE","76","CC","60"
9255 DATA "FE","CD","30","FE","3E","0C"
9260 DATA "CD","FC","FD","18","E3","C5"
9265 DATA "F5","CD","2B","2D","CD","32"
9270 DATA "20","3E","16","D7","F1","D7"
9270 DATA "20","3E","16","D7","F1","D7"
9275 DATA "3E","1C","D7","C1","78","CD"
9280 DATA "E7","FD","79","F5","1F","1F"
9285 REM
9290 REM    DETECT TERZA PARTE
9295 REM
```

```
9300 DATA "1F","1F","CD","F0","FD","F1"
9305 DATA "E6","0F","C6","30","FE","3A"
9310 DATA "38","02","C6","07","D7","C9"
9315 DATA "CD","1A","FE","FD","CB","01"
9320 DATA "AE","CD","4C","FE","06","19"
9325 DATA "76","10","FD","FD","CB","01"
9330 DATA "6E","28","F2","CD","1A","FE"
9335 DATA "3A","08","5C","F6","20","C9"
9340 DATA "01","07","07","F5","C5","41"
9345 DATA "CD","00","0E","C1","10","F8"
9350 DATA "F1","C9","CD","6B","0D","3E"
9355 DATA "09","CD","47","FE","21","B0"
9360 DATA "00","11","20","00","C3","B5"
9365 DATA "03","21","5E","9F","ED","5B"
9370 DATA "0B","5B","A7","ED","52","D0"
9375 DATA "C1","18","E9","F5","CD","30"
9380 DATA "FE","F1","F5","CD","58","FE"
9385 REM
9390 REM   DETECT QUARTA PARTE
9395 REM
9400 DATA "11","6F","FE","CD","0A","0C"
9405 DATA "F1","C9","F5","3E","02","CD"
9410 DATA "01","16","F1","C9","3E","02"
9415 DATA "32","74","5C","DD","21","00"
9420 DATA "5B","21","C0","5D","C3","5A"
9425 DATA "07","80","16","05","00","11"
9430 DATA "06","52","65","67","69","73"
9435 DATA "74","72","61","74","6F","20"
9440 DATA "63","6F","6D","65","20","46"
9445 DATA "69","6C","65","20","64","69"
9450 DATA "20","54","49","50","4F","20"
9455 DATA "2D","3E","20","8D","16","0A"
9460 DATA "00","11","04","4C","75","6E"
9465 DATA "67","68","65","7A","7A","61"
9470 DATA "20","46","49","4C","45","20"
9475 DATA "2D","20","3D","A0","11","03"
9480 DATA "16","0D","00","4C","69","6E"
9485 REM
9490 REM   DETECT QUINTA PARTE
9495 REM
9500 DATA "65","61","20","41","75","74"
9505 DATA "6F","20","45","78","65","63"
9510 DATA "2D","20","3D","A0","11","03"
```

```
9515 DATA "16","0D","00","4C","6F","63"
9520 DATA "61","7A","69","6F","6E","65"
9525 DATA "20","49","6E","69","7A","69"
9530 DATA "6F","20","3D","A0","11","00"
9535 DATA "12","01","4E","45","53","53"
9540 DATA "55","4E","41","A0","16","08"
9545 DATA "13","11","02","44","65","63"
9550 DATA "3A","44","41","54","49","3A"
9555 DATA "48","65","78","A0","16","10"
9560 DATA "00","12","01","11","04","2A"
9565 DATA "43","61","72","69","63","61"
9570 DATA "6D","65","6E","74","6F","20"
9575 DATA "65","20","52","69","6C","6F"
9580 DATA "63","61","7A","69","6F","6E"
9585 REM
9590 REM    DETECT SESTA PARTE
9595 REM
9600 DATA "65","20","46","49","4C","45"
9605 DATA "AA","16","10","00","11","05"
9610 DATA "15","01","2D","53","6F","73"
9615 DATA "74","69","74","75","69","73"
9620 DATA "63","69","20","6C","61","20"
9625 DATA "63","61","73","73","65","74"
9630 DATA "74","61","20","2D","2D","2D"
9635 DATA "2D","2D","2D","AD","16","10"
9640 DATA "00","11","06","15","01","56"
9645 DATA "65","72","69","66","79","20"
9650 DATA "2A","2A","20","53","61","76"
9655 DATA "65","20","2A","2A","20","4C"
9660 DATA "6F","61","64","20","3F","20"
9665 DATA "20","56","2F","53","2F","4C"
9670 DATA "A0","12","01","11","02","3C"
9675 DATA "3C","44","45","54","45","43"
9680 DATA "54","20","42","61","63","6B"
9685 REM
9690 REM    DETECT SETTIMA PARTE
9695 REM
9700 DATA "75","70","2A","20","56","20"
9705 DATA "31","2E","30","20","30","39"
9710 DATA "2F","31","39","38","33","3E"
9715 DATA "3E","14","01","11","03","53"
9720 DATA "69","73","74","65","6D","61"
9725 DATA "20","64","69","20","42","41"
```

```
9730 DATA "43","4B","55","50","20","64"
9735 DATA "65","69","20","46","49","4C"
9740 DATA "45","53","20","2D","2B","2D"
9745 DATA "AB","16","03","06","11","05"
9750 DATA "12","01","50","52","4F","47"
9755 DATA "52","41","4D","4D","41","12"
9760 DATA "00","3A","11","06","A0","16"
9765 DATA "10","00","11","02","12","01"
9770 DATA "53","61","6C","76","61","74"
9775 DATA "61","67","67","69","6F","20"
9780 DATA "46","49","4C","45","20","52"
9785 REM
9790 REM     DETECT OTTAVA PARTE
9795 REM
9800 DATA "69","6C","6F","63","61","74"
9805 DATA "6F","2E","2E","2E","2E","2E"
9810 DATA "2E","AE","16","10","0E","15"
9815 DATA "01","4F","2E","6B","AE","00"
9820 DATA "**"
```

ASS. DETECT 1.0

Questo è il listato assembler del programma di backup DETECT 1.0. È molto lungo per cui potrebbe essere noioso commentarlo linea per linea. Tuttavia nel listato ogni gruppo di istruzioni è preceduto da un breve commento per indicare ciò che esse fanno.

Vorrei invece soffermarmi sulle subroutine della ROM utilizzate. Esse sono elencate nelle prime righe del listato assembler come routine di sistema e sono:

0D6B CLS — Clear screen, pulisce il video.

0E00 SCLL — Scroll: il registro B dello Z-80 contiene il numero di linee che dal basso si vogliono far "Scrollare" verso l'alto.

03B5 BELL — In base al contenuto dei registri HL e DE si emette un suono.

0556 LOAD Carica DE bytes memorizzandoli dalla locazione puntata da IX in poi.

04C2 SAVE Salva DE bytes a partire dalla locazione puntata da IX in poi.

Un file registrato su cassetta si compone di due parti:

1) una parte iniziale, 17 bytes, che contiene l'informazione sul tipo di file in questione;
2) una seconda parte che costituisce il programma vero e proprio.

Per usare le due ultime subroutine, SAVE e LOAD, bisogna caricare il registro A con 0 per la prima parte, e con FF esadecimale per la seconda e settare il flag di carry e ovviamente caricare i registri DE e IX con i valori opportuni.

075A TAPE È la routine di uso generale per il registratore la cui funzione dipende dal valore contenuto nella variabile di sistema TADD (5C74 H) e cioè:

TADD	Funzione di TAPE
0	SAVE
1	LOAD
2	VERIFY
3	MERGE

In ogni caso i registri HL e IX devono essere caricati prima di accedere a TAPE e cioè:

— il registro HL deve puntare a una zona di RAM nella quale si andrà a caricare il file con funzione di LOAD e MERGE, o dalla quale si prelevano i dati da salvare o da verificare;
— il registro IX deve puntare a una zona di RAM che contiene gli attributi sul tipo di file sul quale le 4 funzioni precedenti devono operare:

LOCAZIONE CONTENUTO

IX+00 tipo di file
0 = BASIC
1 = array alfanumerico
2 = array numerico
3 = byte

IX+01	nome del file, FF se il
IX+0A	il file non ha titolo
IX+0B	lunghezza
IX+0C	file
IX+0D	numero di linea
IX+0E	se il file è Basic. Locazione da cui si deve salvare o caricare se il file è binario. Codice che identifica il tipo di array
IX+0F	lunghezza area variabili
IX+10	Basic

203C MSGG immette sul canale corrente (video, stampante...) una stringa di BC caratteri memorizzata all'indirizzo DE

0C0A STR$ questa routine scrive un messaggio sul canale selezionato preso da una lista posta in memoria. Per usare la routine DE deve contenere l'indirizzo di partenza della lista, A il numero del messaggio che si richiede. Ogni messaggio deve avere l'ultimo carattere con il bit 7 posto a uno

2D2B STCK pone sullo stack del Basic il numero contenuto in BC

2032 PTST scrive sul canale selezionato il contenuto di tale stack

1601 OPEN apre il canale il cui numero è posto in A.

```
FD1E          0000        ORG   0FD1EH
              0010 ;
              0020 ;            "Routine"
              0030 ;            "Sistema"
              0040 ;
0D6B          0050 CLS    EQU   0D6BH
0E00          0060 SCLL   EQU   0E00H
03B5          0070 BELL   EQU   03B5H
0556          0080 LOAD   EQU   0556H
075A          0090 TAPE   EQU   075AH
203C          0100 MSGG   EQU   203CH
0C0A          0110 STR$   EQU   0C0AH
```

```
2D2B            0120 STCK   EQU  2D2BH
2032            0130 PTST   EQU  2032H
1601            0140 OPEN   EQU  1601H
                0150 ;
                0160 ;           "Variab."
                0170 ;
5C08            0180 LASK   EQU  5C08H
5C74            0190 TADD   EQU  5C74H
5B00            0200 FILE   EQU  5B00H
09A2            0210 CHR1   EQU  09A2H
5DC0            0220 RELC   EQU  24000
                0230 ;
                0240 ;           "DETECT 1.
0"
                0250 ;
FD1E CD28FE     0260 INIZ   CALL CLSC
FD21 3E06       0270        LD   A,6
FD23 CD47FE     0280        CALL PRING
FD26 AF         0290        XOR  A
FD27 37         0300        SCF
FD28 DD21005B   0310        LD   IX,FILE
FD2C 111100     0320        LD   DE,11H
FD2F CD5605     0330        CALL LOAD
FD32 30EA       0340        JR   NC,INIZ
                0350 ;
                0360 ;           "MSG..PROG
RAMMA"
                0370 ;
FD34 3E0A       0380        LD   A,10
FD36 CD47FE     0390        CALL PRING
                0400 ;
                0410 ;           "MSG..NOME
 FILE"
                0420 ;
FD39 11015B     0430        LD   DE,FILE+1
FD3C 1A         0440        LD   A,(DE)
FD3D FEFF       0450        CP   0FFH
FD3F 2806       0460        JR   Z,UNO
FD41 010A00     0470        LD   BC,10
FD44 CD3C20     0480        CALL MSGG
                0490 ;
                0500 ;           "MSG..1 ME
SSAGGIO"
```

```
                0510 ;
FD47 AF         0520 UNO    XOR  A
FD48 CD47FE     0530        CALL PRING
                0540 ;
                0550 ;           "MSG..TIPO
 FILE"
                0560 ;
FD4B 3E11       0570        LD   A,11H
FD4D D7         0580        RST  10H
FD4E 3A005B     0590        LD   A,(FILE)
FD51 F5         0600        PUSH AF
FD52 D7         0610        RST  10H
FD53 F1         0620        POP  AF
FD54 F5         0630        PUSH AF
FD55 11A209     0640        LD   DE,CHR1
FD58 CD0A0C     0650        CALL STR$
                0660 ;
                0670 ;           "MSG..DATI
"
                0680 ;
FD5B 3E05       0690        LD   A,5
FD5D CD47FE     0700        CALL PRING
                0710 ;
                0720 ;           "MSG..LUNG
HEZZA"
                0730 ;
FD60 3E01       0740        LD   A,1
FD62 CD47FE     0750        CALL PRING
                0760 ;
                0770 ;           "PRINT NUM
.BYTES"
                0780 ;
FD65 ED4B0B5B 0790          LD   BC,(FILE+0
BH)
FD69 3E0A       0800        LD   A,10
FD6B CDD1FD     0810        CALL PTBC
                0820 ;
                0830 ;           "MSG..Loca
zione inizio"
                0840 ;           "o Linea d
i AUTo-EXEC"
                0850 ;
FD6E F1         0860        POP  AF
```

```
FD6F FE00     0870           CP   0
FD71 2014     0880           JR   NZ,BYTE
              0890 ;
              0900 ;              "MSG..Line
a AUTO-EXEC"
              0910 ;
FD73 3E02     0920           LD   A,2
FD75 CD47FE   0930           CALL PRING
FD78 ED4B0D5B 0940           LD   BC,(FILE+0
DH)
FD7C CB78     0950           BIT  7,B
FD7E 2812     0960           JR   Z,PBC
              0970 ;
              0980 ;              "MSG..NESS
UN AUTO-EXEC"
              0990 ;
FD80 3E04     1000           LD   A,4
FD82 C447FE   1010           CALL NZ,PRING
FD85 1810     1020           JR   CONT
FD87 FE03     1030 BYTE      CP   3
FD89 200C     1040           JR   NZ,CONT
              1050 ;
              1060 ;              "MSG..LOCA
ZIONE INIZIO"
              1070 ;
FD8B CD47FE   1080           CALL PRING
FD8E ED4B0D5B 1090           LD   BC,(FILE+0
FD92 3E0D     1100 PBC       LD   A,13
FD94 CDD1FD   1110           CALL PTBC
              1120 ;
              1130 ;              "TEST sull
a LUNGHEZZA"
              1140 ;
FD97 CD39FE   1150 CONT      CALL TEST
              1160 ;
              1170 ;              "CARICAMEN
TO FILE"
              1180 ;
FD9A DD21C05D 1190           LD   IX,RELC
FD9E 3EFF     1200           LD   A,0FFH
FDA0 37       1210           SCF
FDA1 CD5605   1220           CALL LOAD
FDA4 30F1     1230           JR   NC,CONT
```

```
                1240 ;
                1250 ;              "MSG..per
la CASSETTA"
                1260 ;
FDA6 3E07       1270          LD   A,7
FDA8 CDFCFD     1280          CALL WAIT
                1290 ;
                1300 ;              "MSG..SALV
ATAGGIO FILE"
                1310 ;
FDAB 3E0B       1320 SALV     LD   A,11
FDAD CD4CFE     1330          CALL PRINT
                1340 ;
                1350 ;              "Salvatagg
io FILE"
                1360 ;
FDB0 AF         1370          XOR  A
FDB1 CD62FE     1380          CALL TAPS
                1390 ;
                1400 ;              "MSG..SALV
.o VERIFICA"
                1410 ;              "          L
OAD         "
FDB4 3E08       1420 ASK      LD   A,8
FDB6 CDFCFD     1430          CALL WAIT
FDB9 FE6C       1440          CP   "l"
FDBB CA1EFD     1450          JP   Z,INIZ
FDBE FE73       1460          CP   "s"
FDC0 28E9       1470          JR   Z,SALV
FDC2 FE76       1480          CP   "v"
FDC4 CC60FE     1490          CALL Z,TAPV
FDC7 CD30FE     1500          CALL RING
FDCA 3E0C       1510          LD   A,12
FDCC CDFCFD     1520          CALL WAIT
FDCF 18E3       1530          JR   ASK
                1540 ;
                1550 ;
                1560 ;              "Sub STAMP
A BC in HEX"
                1570 ;              "    e in
DECIMALE    "
                1580 ;
FDD1 C5         1590 PTBC     PUSH BC
```

```
FDD2 F5         1600          PUSH AF
                1610 ;
                1620 ;              "PUSH BC s
ullo STACK"
                1630 ;              "Matematic
o "
                1640 ;
FDD3 CD2B2D     1650          CALL STCK
                1660 ;
                1670 ;              "STAMPA il
 CONTENUTO"
                1680 ;              "STACK MAT
EMATICO"
                1690 ;
FDD6 CD3220     1700          CALL PTST
                1710 ;
                1720 ;              "TABULAZIO
NE "
                1730 ;
FDD9 3E16       1740          LD   A,16H
FDDB D7         1750          RST  10H
FDDC F1         1760          POP  AF
FDDD D7         1770          RST  10H
FDDE 3E1C       1780          LD   A,1CH
FDE0 D7         1790          RST  10H
                1800 ;
                1810 ;              "STAMPA BC
 in HEX"
                1820 ;
FDE1 C1         1830          POP  BC
FDE2 78         1840 ESA      LD   A,B
FDE3 CDE7FD     1850          CALL CONV
FDE6 79         1860          LD   A,C
FDE7 F5         1870 CONV     PUSH AF
FDE8 1F         1880          RRA
FDE9 1F         1890          RRA
FDEA 1F         1900          RRA
FDEB 1F         1910          RRA
FDEC CDF0FD     1920          CALL NIBLE
FDEF F1         1930          POP  AF
FDF0 E60F       1940 NIBLE    AND  0FH
FDF2 C630       1950          ADD  30H
FDF4 FE3A       1960          CP   3AH
```

```
FDF6 3802       1970            JR   C,OUT1
FDF8 C607       1980            ADD  7
FDFA D7         1990 OUT1       RST  10H
FDFB C9         2000            RET
                2010 ;
                2020 ;               "Sub Wait"
                2030 ;
FDFC CD1AFE     2040 WAIT       CALL SCBL
FDFF FDCB01AE   2050            RES  5,(IY+1)
FE03 CD4CFE     2060 BLINK      CALL PRINT
FE06 0619       2070            LD   B,25
FE08 76         2080 ALT        HALT
FE09 10FD       2090            DJNZ ALT
FE0B FDCB016E   2100            BIT  5,(IY+1)
FE0F 28F2       2110            JR   Z,BLINK
FE11 CD1AFE     2120            CALL SCBL
FE14 3A085C     2130            LD   A,(LASK)
FE17 F620       2140            OR   20H
FE19 C9         2150            RET
                2160 ;
                2170 ;               "Sub SCROL
L B Linee"
                2180 ;               "    per B
volte  "
                2190 ;
FE1A 010707     2200 SCBL       LD   BC,0707H
FE1D F5         2210            PUSH AF
FE1E C5         2220 SC         PUSH BC
FE1F 41         2230            LD   B,C
FE20 CD000E     2240            CALL SCLL
FE23 C1         2250            POP  BC
FE24 10F8       2260            DJNZ SC
FE26 F1         2270            POP  AF
FE27 C9         2280            RET
                2290 ;
                2300 ;               "Sub Clear
 Screen"
                2310 ;
FE28 CD6B0D     2320 CLSC       CALL CLS
FE2B 3E09       2330            LD   A,9
FE2D CD47FE     2340            CALL PRING
FE30 21B000     2350 RING       LD   HL,00B0H
FE33 112000     2360            LD   DE,0020H
```

```
FE36 C3B503    2370            JP    BELL
               2380 ;
               2390 ;                "Sub TEST"
               2400 ;
9F5E           2410 COST  DEFL  INIZ-RELC
FE39 215E9F    2420 TEST  LD    HL,COST
FE3C ED5B0B5B  2430       LD    DE,(FILE+0
BH)
FE40 A7        2440       AND   A
FE41 ED52      2450       SBC   HL,DE
FE43 D0        2460       RET   NC
FE44 C1        2470       POP   BC
FE45 18E9      2480       JR    RING
               2490 ;
               2500 ;                "Sub PRINT
"
               2510 ;
FE47 F5        2520 PRING PUSH  AF
FE48 CD30FE    2530       CALL  RING
FE4B F1        2540       POP   AF
FE4C F5        2550 PRINT PUSH  AF
FE4D CD58FE    2560       CALL  CHAN
FE50 116FFE    2570       LD    DE,CHRS
FE53 CD0A0C    2580       CALL  STR$
FE56 F1        2590       POP   AF
FE57 C9        2600       RET
FE58 F5        2610 CHAN  PUSH  AF
FE59 3E02      2620       LD    A,2
FE5B CD0116    2630       CALL  OPEN
FE5E F1        2640       POP   AF
FE5F C9        2650       RET
               2660 ;
               2670 ;                "Sub SAVE
o VERIFY"
               2680 ;
FE60 3E02      2690 TAPV  LD    A,2
FE62 32745C    2700 TAPS  LD    (TADD),A
FE65 DD21005B  2710       LD    IX,FILE
FE69 21C05D    2720       LD    HL,RELC
FE6C C35A07    2730       JP    TAPE
               2740 ;
               2750 ;                "Messaggi"
               2760 ;
```

```
                  2770 ;            "---------
0--------"
FE6F 80           2780 CHRS  DEFB 80H
FE70 16           2790       DEFB 16H
FE71 0500         2800       DEFW 0005H
FE73 1106         2810       DEFW 0611H
FE75              2820       DEFM "Registrat
o come "
FE85              2830       DEFM "File di T
IPO -> "
FE95 8D           2840       DEFB 08DH
                  2850 ;            "---------
1--------"
FE96 16           2860       DEFB 16H
FE97 0A00         2870       DEFW 000AH
FE99 1104         2880       DEFW 0411H
FE9B              2890       DEFM "Lunghezza
 FILE - ="
FEAD A0           2900       DEFB 0A0H
                  2910 ;            "---------
2--------"
FEAE 1103         2920       DEFW 0311H
FEB0 16           2930       DEFB 16H
FEB1 0D00         2940       DEFW 000DH
FEB3              2950       DEFM "Linea Aut
o Exec- ="
FEC5 A0           2960       DEFB 0A0H
                  2970 ;            "---------
3--------"
FEC6 1103         2980       DEFW 0311H
FEC8 16           2990       DEFB 16H
FEC9 0D00         3000       DEFW 000DH
FECB              3010       DEFM "Locazione
 Inizio ="
FEDD A0           3020       DEFB 0A0H
                  3030 ;            "---------
4--------"
FEDE 1100         3040       DEFW 0011H
FEE0 1201         3050       DEFW 0112H
FEE2              3060       DEFM "NESSUNA"
FEE9 A0           3070       DEFB 0A0H
                  3080 ;            "---------
5--------"
```

```
FEEA 16        3090       DEFB 16H
FEEB 0813      3100       DEFW 1308H
FEED 1102      3110       DEFW 0211H
FEEF           3120       DEFM "Dec:DATI:
Hex"
FEFB A0        3130       DEFB 0A0H
               3140 ;          "---------
6---------"
FEFC 16        3150       DEFB 16H
FEFD 1000      3160       DEFW 0010H
FEFF 1201      3170       DEFW 0112H
FF01 1104      3180       DEFW 0411H
FF03           3190       DEFM "*Caricame
nto e Ril"
FF15           3200       DEFM "ocazione
FILE"
FF22 AA        3210       DEFB "*"+80H
               3220 ;          "---------
7---------"
FF23 16        3230       DEFB 16H
FF24 1000      3240       DEFW 0010H
FF26 1105      3250       DEFW 0511H
FF28 1501      3260       DEFW 0115H
FF2A           3270       DEFM "-Sostitui
sci la ca"
FF3C           3280       DEFM "ssetta --
----"
FF49 AD        3290       DEFB "-"+80H
               3300 ;          "---------
8---------"
FF4A 16        3310       DEFB 16H
FF4B 1000      3320       DEFW 0010H
FF4D 1106      3330       DEFW 0611H
FF4F 1501      3340       DEFW 0115H
FF51           3350       DEFM "Verify **
 Save ** "
FF63           3360       DEFM "Load ?  V
/S/L"
FF70 A0        3370       DEFB 0A0H
               3380 ;          "---------
9---------"
FF71 1201      3390       DEFW 0112H
FF73 1102      3400       DEFW 0211H
```

```
FF75            3410        DEFM "<<DETECT
Backup*"
FF85            3420        DEFM " V 1.0 09
/1983>>"
FF95 1401       3430        DEFW 0114H
FF97 1103       3440        DEFW 0311H
FF99            3450        DEFM "Sistema d
i BACKUP "
FFAB            3460        DEFM "dei FILES
 -+-"
FFB8 AB         3470        DEFB "+"+80H
                3480 ;           "--------1
0--------"
FFB9 16         3490        DEFB 16H
FFBA 0306       3500        DEFW 0603H
FFBC 1105       3510        DEFW 0511H
FFBE 1201       3520        DEFW 0112H
FFC0            3530        DEFM "PROGRAMMA
"
FFC9 1200       3540        DEFW 0012H
FFCB 3A         3550        DEFB ":"
FFCC 1106       3560        DEFW 0611H
FFCE A0         3570        DEFB 0A0H
                3580 ;           "--------1
1--------"
FFCF 16         3590        DEFB 16H
FFD0 1000       3600        DEFW 0010H
FFD2 1102       3610        DEFW 0211H
FFD4 1201       3620        DEFW 0112H
FFD6            3630        DEFM "Salvatagg
io FILE R"
FFE8            3640        DEFM "ilocato..
...."
FFF5 AE         3650        DEFB "."+80H
                3660 ;           "--------1
2--------"
FFF6 16         3670        DEFB 16H
FFF7 100E       3680        DEFW 0E10H
FFF9 1501       3690        DEFW 0115H
FFFB            3700        DEFM "O.k"
FFFE AE         3710 LAST1  DEFB "."+80H
                3720        END
LAST1  FFFE
```

```
CHRS    FE6F
TAPS    FE62
TAPY    FE60
CHAN    FE58
PRINT   FE4C
PRING   FE47
TEST    FE39
COST   *9F5E
RING    FE30
CLSC    FE28
SC      FE1E
SCBL    FE1A
ALT     FE08
BLINK   FE03
WAIT    FDFC
OUT1    FDFA
NIBLE   FDF0
CONV    FDE7
ESA     FDE2
PTBC    FDD1
ASK     FDB4
SALV    FDAB
CONT    FD97
PBC     FD92
BYTE    FD87
UNO     FD47
INIZ    FD1E
RELC    5DC0
CHR1    09A2
FILE    5B00
TADD    5C74
LASK    5C08
OPEN    1601
PTST    2032
STCK    2D2B
STR#    0C0A
MSGG    203C
TAPE    075A
LOAD    0556
BELL    03B5
SCLL    0E00
CLS     0D6B
#       6062
```

DETECT –S

DETECT 1.0 presentato altrove in queste pagine è un programma per il backup dei nastri ed è in grado di dare tutte le informazioni relative ad un file registrato su una cassetta. Queste informazioni sono presentate su video nei due formati numerici, decimale ed esadecimale.

Tuttavia tale programma ha una limitazione:

– non sempre è possibile duplicare un file perché troppo lungo, per cui DETECT 1.0 ritorna al BASIC rifiutandosi di caricare un solo byte.

Fortunatamente questa evenienza capita assai di rado perché è abbastanza difficile, ma non improbabile, incontrare file più lunghi di 40.000 bytes.
Quando ciò capita è possibile risolvere il problema usando DETECT –S. Il programma che segue non è altro che il caricatore del linguaggio macchina che realizza il backup.
Infatti dopo aver dato il RUN il computer darà il solito messaggio che precede ogni registrazione. Bisogna quindi disporre di una cassetta nuova su cui registrare il DETECT –S in L/M.

L'uso poi di quest'oggetto è abbastanza semplice anche se bisogna prestare molta attenzione e in generale occorre dare queste istruzioni:

1) spegnere e accendere il computer
2) CLEAR 23999
3) LOAD "DETECT –S" CODE 23400
4) RANDOMIZE USR 23400

Dopo quest'ultimo comando il computer sul video non mostra niente ma, se tutto procede bene, carica il primo file da duplicare che trova su cassetta. Ciò viene indicato dalle classiche righe variopinte sul BORDER.
Quando comparirà sulle ultime due righe dello schermo il classico messaggio "Start Tape and Press any Key", il computer avrà rilocato tutto il file da duplicare a partire dalla locazione 24000 in poi. Inserendo una cassetta e premendo un tasto il file viene salvato.

```
1000 REM
1005 REM PROGRAMMA  DI BACKUP
1010 REM MAX.41.5K BYTES FILE
1015 REM
1020 REM  HEX-CODE Caricatore
1025 REM
1030 CLEAR 31999: LET a=32000
1035 PRINT AT 10,11;"ATTENDERE"
1040 FOR i=0 TO 32
1045 BEEP .01,65: READ a$
1050 LET h=CODE a$(1)-48
1055 LET h=h-7*(a$(1)>"@")
1060 LET l=CODE a$(2)-48
1065 LET l=l-7*(a$(2)>"@")
1070 POKE a+i,16*h+l: NEXT i
1075 SAVE "DETECT -S"CODE a,i
1080 REM
1085 REM  DATI PROGRAMMA L.M.
1090 REM
1095 REM        DETECT -S
1100 REM
1105 DATA "AF","DD","21","00"
1110 DATA "5B","11","11","00"
1115 DATA "37","CD","56","05"
1120 DATA "DD","21","C0","5D"
1125 DATA "ED","5B","0B","5B"
1130 DATA "CD","6D","08","21"
1135 DATA "C0","5D","DD","21"
1140 DATA "00","5B","C3","70"
1145 DATA "09"
```

ASS. DETECT –S

Il listato seguente è l'assembler della routine descritta nell'omonimo programma in BASIC. È molto breve e si nota che è rilocabile in ogni zona della memoria.

Genericamente verrà caricata nel buffer di stampante a partire dalla locazione 23400.

Questa si basa su 3 subroutine nella ROM dello Spectrum che verranno ora brevemente descritte.

Commento:

LINEE da 160 a 200:

prima di accedere alla routine LOAD (0556 HEX) si devono dare dei valori opportuni ai registri IX, DE e A dello Z-80:

- i registri A uguale a 0, DE uguale a 17 indicano al computer di caricare i primi 17 bytes della parte iniziale, l'HEADER, del file memorizzandoli a partire dalla locazione puntata dal registro IX (i primi 17 bytes del buffer di stampante in questa applicazione).

LINEE da 210 a 230:

- la routine LOAT (086D HEX) è più o meno identica alla precedente e serve per caricare l'intero corpo di un file. Per accedervi bisogna specificare nel registro IX l'indirizzo della locazione (24000 nel caso particolare) dalla quale cominciare a caricare i bytes nel numero specificato nei registri DE. Tale numero per un dato file lo si ricava dalle informazioni ricevute caricando l'HEADER.

LINEE da 240 a 260:

- si salta alla routine SAVE (0970 HEX) dopo aver specificato in HL l'indirizzo di partenza dei bytes da salvare (24000) e in IX l'indirizzo ove risiedono le informazioni (il contenuto dell'HEADER) relative al file che si desidera duplicare.
  Con qualche modifica è possibile caricare (e quindi duplicare) più di 41.5k bytes agendo sulla variabile RELO, attualmente uguale a 24000, abbassandone il valore. In ambiente BASIC prima di lanciare la subroutine occorre posizionare la RAMTOP in accordo con il valore di RELO impostato.

```
6062            0000          ORG  #
                0010  ;
                0020  ;
                0030  ;            "Routine"
                0040  ;            "Sistema"
                0050  ;
```

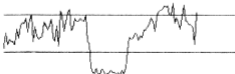
```
0556              0060 LOAD   EQU  0556H
086D              0070 LOAT   EQU  086DH
0970              0080 SAVE   EQU  0970H
                  0090 ;
                  0100 ;           "Variab."
                  0110 ;
5B00              0120 INFO   EQU  5B00H
5DC0              0130 RELO   EQU  24000
                  0140 ;
                  0150 ;
6062 AF           0160 INIZ   XOR  A
6063 DD21005B     0170        LD   IX,INFO
6067 111100       0180        LD   DE,11H
606A 37           0190        SCF
606B CD5605       0200        CALL LOAD
606E DD21C05D     0210        LD   IX,RELO
6072 ED5B0B5B     0220        LD   DE,(INFO+0
BH)
6076 CD6D08       0230        CALL LOAT
6079 21C05D       0240        LD   HL,RELO
607C DD21005B     0250        LD   IX,INFO
6080 C37009       0260        JP   SAVE
                  0270        END

INIZ    6062
RELO    5DC0
INFO    5B00
SAVE    0970
LOAT    086D
LOAD    0556
#       6062
```

HEAR THE MIC

Questa routinetta di pochi byte non serve a granché, tuttavia è di grande effetto scenico perché se rileva la presenza di un segnale alla presa EAR dello Spectrum ne da una rappresentazione grafica sullo schermo. Il grafico che si ottiene è più legato al livello del segnale che alla sua frequenza per

cui una valida applicazione può essere il test del livello di registrazione delle cassette di personale incisione.

```
1000 REM
1005 REM    HEAR THE MIC
1010 REM LETTURA PRESA EAR
1015 REM E VISUALIZZAZIONE
1020 REM
1025 CLEAR 31999: GO SUB 9000
1030 LET x=0: LET y=USR 32000
1035 FOR n=0 TO 255 STEP 2
1040 LET l=USR 32000: PLOT n,l
1045 DRAW x-n,-l+y: LET x=n
1050 LET y=l: NEXT n
1055 CLS : GO TO 1030
1060 REM
1065 REM HEX-CODE CARICATORE
1070 REM
9000 FOR p=32000 TO 32016
9005 READ a$
9010 LET h=CODE a$(1)-48
9015 LET h=h-7*(a$(1)>"@")
9020 LET l=CODE a$(2)-48
9025 LET l=l-7*(a$(2)>"@")
9030 POKE p,16*h+l
9035 NEXT p: RETURN
9085 REM
9090 REM    SUB HEAR THE MIC
9095 REM
9100 DATA "F3","01","00","AF"
9105 DATA "3E","7F","DB","FE"
9110 DATA "CB","77","20","01"
9115 DATA "0C","10","F5","FB"
9120 DATA "C9"
```

ASS. HEAR THE MIC

Data la semplicità di questa routine, c'é poco da dire sul listato assembler. Si rileva il dato alla porta 7FFE H, si controlla la presenza o meno di un segnale (BIT 6,A) e nel caso si incrementa il registro C. Il ciclo è svolto per 175 volte (registro B) e il valore ritornato al Basic è il valore associato alla coppia BC con B sempre a zero.

```
7D00            0000        ORG   32000
                0010 ;
                0020 ;            "HEAR"
                0030 ;            "THE "
                0040 ;            "MIC."
                0050 ;
                0060 ;            "Variab."
                0070 ;
AF00            0080 MAX    EQU   0AF00H
                0090 ;
                0100 ;
7D00 F3         0110 IRQD   DI
7D01 0100AF     0120        LD    BC,MAX
7D04 3E7F       0130 TEST   LD    A,7FH
7D06 DBFE       0140        IN    A,(0FEH)
7D08 CB77       0150        BIT   6,A
7D0A 2001       0160        JR    NZ,NSET
7D0C 0C         0170        INC   C
7D0D 10F5       0180 NSET   DJNZ  TEST
7D0F FB         0190        EI
7D10 C9         0200        RET
                0210        END
NSET    7D0D
TEST    7D04
IRQD    7D00
MAX     AF00
#       6062
```

IMPULSE

Questa routine a differenza della precedente è più utile. Sente la presenza di un segnale impulsivo all'ingresso e restituisce al Basic un valore inversamente proporzionale alla sua durata.

Anche in questo caso se ne da una rappresentazione grafica su video.

Il suo uso può essere quello di rilevare la variazione in frequenza (con molta approssimazione) che può essere presente, causa distorsioni o altro, nel leggere l'header, cioè la parte iniziale di ogni programma registrato su nastro. Questo è infatti preceduto da una nota fissa facilmente rilevabile da questo programma. In ogni caso provate a collegare alla presa EAR una radiolina e osservate l'effetto.

```
1000 REM
1005 REM MISURA DI UN IMPULSO
1010 REM  E RAPPRESENTAZIONE
1015 REM         GRAFICA
1020 REM
1025 GO TO 9000: REM RAM MEMO
1030 DEF FN c(y)=INT (y/30)
1035 LET a=0: LET b=a: LET s=2
1040 CLS : PLOT 0,80: DRAW 255,0
1045 PLOT 0,20: DRAW 255,0
1050 FOR x=0 TO 255 STEP s
1055 LET y=USR 32000: PLOT x,y
1060 DRAW  INK FN c(y);a-x,b-y
1065 LET a=x: LET b=y: NEXT x
1070 LET a=0: GO TO 1040
9000 REM
9005 REM  HEX-CODE Caricatore
9010 REM
9030 CLEAR 31999: LET a=32000
9035 READ a$
9040 IF a$="**" THEN  GO TO 1030
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 POKE a,16*h+l: LET a=a+1
9070 GO TO 9035
```

```
9075 REM
9080 REM  DATI PROGRAMMA L.M.
9085 REM
9090 REM       SUB IMPULSE
9095 REM
9100 DATA "F3","01","22","00"
9105 DATA "CD","E3","05","30"
9110 DATA "F8","01","02","00"
9115 DATA "CD","E3","05","30"
9120 DATA "F0","78","06","00"
9125 DATA "FE","B0","38","02"
9130 DATA "3E","AF","4F","FB"
9135 DATA "C9","**"
```

ASS. IMPULSE

La base fondamentale di questo programma è una subroutine mappata in ROM all'indirizzo 05E3 H chiamata EDGE che presiede alla rilevazione degli impulsi all'ingresso EAR.

Questa costituisce la parte più importante per le operazioni di LOAD - VERIFY e ha due punti di ingresso:

EDGE–1 = 05E3

EDGE–2 = 05E7

È noto che i dati spediti al generico registratore a cassette che tutti noi abbiamo al seguito del nostro Spectrum, sono visti come una serie di impulsi registrati su nastro e come tali possono venire viceversa letti dal computer.

Se potessimo visualizzare la "forma" di questi impulsi essi assomiglierebbero a un "onda quadra". Questa è costituita da un fronte di salita e uno di discesa e il tempo che intercorre fra di essi ne costituisce la durata.

Si esegue EDGE-1 o EDGE-2 con una costante nel registro B e con il tipo di picco che si vuol rilevare nel registro C. Al ritorno da questa subroutine il flag di carry è posto a uno se si è trovato il numero richiesto di "impulsi" nel tempo stabilito in B e la variazione avvenuta in questo registro (B), stabilisce quanto tempo è stato necessario per trovare gli impulsi citati.

Si chiama EDGE-1 quando occorre rilevare la lunghezza di un impulso completo.

Si chiama EDGE-2 quando è necessario misurare il tempo precedente la rilevazione di un "picco".

Spero che risulti abbastanza chiaro che, opportunamente caricando il registro B con un valore, ci si possa sincronizzare su un impulso di durata nota se questo è presente all'ingresso EAR.

Per maggiori informazioni consiglio di consultare:

THE COMPLETE SPECTRUM ROM DISASSEMBLY
di Dr. Ian LOGAN - Dr. Frank O' HARA.

```
7D00            0000         ORG   32000
                0010 ;
                0020 ;             "Routine"
                0030 ;             "Sistema"
                0040 ;
05E3            0050 EDGE    EQU   05E3H
                0060 ;
7D00 F3         0070 IRQD    DI
7D01 012200     0080 SYNC    LD    BC,0022H
7D04 CDE305     0090         CALL  EDGE
7D07 30F8       0100         JR    NC,SYNC
7D09 010200     0110         LD    BC,0002H
7D0C CDE305     0120         CALL  EDGE
7D0F 30F0       0130         JR    NC,SYNC
7D11 78         0140         LD    A,B
7D12 0600       0150         LD    B,0
7D14 FEB0       0160         CP    0B0H
7D16 3802       0170         JR    C,LB
7D18 3EAF       0180         LD    A,0AFH
7D1A 4F         0190 LB      LD    C,A
7D1B FB         0200         EI
```

```
7D1C C9          0210          RET
                 0220          END
LB      7D1A
SYNC    7D01
IRQD    7D00
EDGE    05E3
#       6062
```

PARTE II

LA GRAFICA

ROUTINE DI SPOSTAMENTO (SHIFT)

Se vi servono routine che lavorano sulla pagina grafica dello Spectrum, quanto segue fa per voi. Se invece non vi servono, vi invito comunque a dare un'occhiata, chissà forse tra una riga e l'altra di questi listati può maturare una nuova idea.

Bene cominciamo subito a parlare delle "solite" routine di shift.

Queste però non sono così usuali come si potrebbe pensare, infatti utilizzano una subroutine, già presente nella ROM dello Spectrum, che ho chiamato BITC.

Questa è una routine che calcola l'indirizzo della locazione video nella quale si andrà a scrivere (usualmente con un istruzione di PLOT, DRAW).

SHIFT significa SPOSTAMENTO ed è a volte desiderabile ottenere lo spostamento di un bit, verso destra o sinistra, verso l'alto o il basso di tutto lo schermo. Ciò è stato fatto nelle subroutine seguenti.

Esse funzionano in questo modo:

1) si usa la subroutine BITC mappata in 22AAH in ROM. Calcola gli indirizzi dei byte su cui operare lo spostamento e come tale;

2) per funzionare correttamente occorre specificare nei registri BC dello Z-80 l'ordinata e l'ascissa del punto che individua la locazione video che per prima deve essere elaborata.

   Esempio: la locazione 16384 (4000 in esadecimale) appartenente alla prima colonna video in alto, la si può calcolare specificando per l'ordina-

ta Y il numero 175 (AF in esadecimale), e come ascissa il numero 0 oppure 01,03,04...07. Il numero 08 come ascissa e sempre 175 come ordinata induce BITC a calcolare la locazione 16385 ovvero 4001 esadecimale, che appartiene alla seconda colonna video.
Avete capito il meccanismo?

3) Per quanto detto al punto 2) si intuisce una cosa importante: è possibile spostare in ogni direzione porzioni ben definite dello schermo specificando in:

   XYCO l'ordinata e l'ascissa del punto che appartiene alla locazione da trattare

   BYTE il numero dei byte sulla riga su cui operare

   FINE l'ordinata del punto (riga) sulla quale ci si deve fermare.

4) Ultima ma non meno importante, è inclusa la possibilità di non perdere il bit che "esce" a sinistra o a destra (in alto o in basso). Perciò il contenuto della pagina grafica non si perde mai: ciò che esce da una parte rientra da quella opposta.

SHBDX

Di seguito vengono presentati due programmi che sono ciascuno l'applicazione della routine in L/M omologa.

Sono molto semplici e differiscono l'uno dall'altro solo per poche istruzioni in assembler.

SHBDX da un esempio di come si possa spostare verso destra, pixel per pixel, la prima linea in alto sullo schermo.

```
1000 REM
1005 REM ROUTINE DI SHIFT
1010 REM       A DESTRA
1015 REM
1020 CLEAR 31999: GO SUB 1065
```

```
1025 CLS : PRINT " Questa e'la";
1030 PRINT " routine di SHIFT ";
1035 PRINT "DX"
1040 RANDOMIZE USR 32000
1045 PAUSE 1: GO TO 1040
1050 REM
1055 REM  HEX-CODE Caricatore
1060 REM
1065 LET a=32000
1070 READ a$
1075 IF a$="**" THEN  RETURN
1080 LET h=CODE a$(1)-48
1085 LET h=h-7*(a$(1)>"@")
1090 LET l=CODE a$(2)-48
1095 LET l=l-7*(a$(2)>"@")
1100 POKE a,16*h+l: LET a=a+1
1105 GO TO 1070
1110 REM
1115 REM  DATI PROGRAMMA L.M.
1120 REM
1125 REM        SUB SHBDX
1130 REM
1135 DATA "01","00","AF","C5"
1140 DATA "CD","AA","22","E5"
1145 DATA "06","20","A7","CB"
1150 DATA "1E","23","10","FB"
1155 DATA "E1","30","02","CB"
1160 DATA "FE","C1","05","78"
1165 DATA "FE","A7","20","E7"
1170 DATA "C9","**"
```

ASS. SHBDX (SHIFT A DESTRA)

Questa routine è molto semplice, è rilocabile in qualsiasi zona della memoria, ed è modificabile per ciò che riguarda le tre variabili principali (YXCO, BYTE, FINE), dall'utente al fine di estendere lo spostamento a una o più linee del video o a tutto lo schermo.

Un breve commento per gruppi di linee servirà a chiarirne il funzionamento:

LINEE da 0130 a 0160:

- si carica nella coppia di registri B e C dello Z80 l'ordinata e l'ascissa del punto che individua la locazione desiderata (ce ne sono 8 di questi punti per gli 8 valori che X può assumere per ciascun byte del video, se ne prende uno solo). Si salva BC sullo stack, si calcola l'indirizzo di questo byte chiamando la routine BITC che deposita tale valore nella coppia di registri HL. Quindi si salva sullo stack HL.

LINEE da 0170 a 0240:

- in questa fase il registro B contiene il numero di byte che su ogni linea si intende elaborare. Perciò costituisce il numero di volte che il ciclo seguente deve essere ripetuto (linee tra 0190 e 0210). Quindi si ripristina il valore dei registri HL e, in base al valore del bit di carry, si elabora il bit 7 della locazione puntata da HL che risulta essere la prima di ogni linea. Questo bit durante il ciclo di rotazione andrebbe perso. In questo modo invece si fa in modo che il bit che esce a destra dello schermo rientri a sinistra e nulla va perso.

LINEE da 0250 a 300:

- si ripristina il valore di BC dallo stack, cioè l'ordinata e l'ascissa del solito punto, si decrementa l'ordinata per passare alla linea successiva, si fa un test di FINE lavoro, quindi si esce o si prosegue il ciclo.

```
7D00          0000          ORG   32000
              0010 ;
              0020 ;              "Routine"
              0030 ;              "Sistema"
              0040 ;
22AA          0050 BITC     EQU   22AAH
              0060 ;
              0070 ;              "Variab."
              0080 ;
AF00          0090 YXCO     EQU   0AF00H
0020          0100 BYTE     EQU   32
00A7          0110 FINE     EQU   0A7H
              0120 ;
```

```
7D00 0100AF      0130 SHBDX LD    BC,YXCO
7D03 C5          0140 LB2   PUSH  BC
7D04 CDAA22      0160       CALL  BITC
7D07 E5          0170       PUSH  HL
7D08 0620        0180       LD    B,BYTE
7D0A A7          0190       AND   A
7D0B CB1E        0200 LB3   RR    (HL)
7D0D 23          0210       INC   HL
7D0E 10FB        0220       DJNZ  LB3
7D10 E1          0230       POP   HL
7D11 3002        0240       JR    NC,LB4
7D13 CBFE        0250       SET   7,(HL)
7D15 C1          0260 LB4   POP   BC
7D16 05          0270       DEC   B
7D17 78          0280       LD    A,B
7D18 FEA7        0290       CP    FINE
7D1A 20E7        0300       JR    NZ,LB2
7D1C C9          0310       RET
                 0320       END

LB4     7D15
LB3     7D0B
LB2     7D03
SHBDX   7D00
FINE    00A7
BYTE    0020
YXCO    AF00
BITC    22AA
#       6062
```

SHBSX

SHBSX sposta pixel per pixel verso sinistra la prima linea dello schermo, così come è stato fatto nel programma precedente.

Tuttavia penso sia abbastanza chiaro che, opportunamente agendo sulle istruzioni in L/M, sia possibile estendere questo spostamento ad altre linee o a particolari porzioni di schermo. A questo scopo è d'obbligo esaminare il listato assembler di ciascuna routine.

```
1000 REM
1005 REM ROUTINE DI SHIFT
1010 REM     A SINISTRA
1015 REM
1020 CLEAR 31999: GO SUB 1065
1025 PRINT " Questa e'la";
1030 PRINT " routine di SHIFT
1035 PRINT "SX"
1040 RANDOMIZE USR 32000
1045 PAUSE 1: GO TO 1040
1050 REM
1055 REM  HEX-CODE Caricatore
1060 REM
1065 LET a=32000
1070 READ a$
1075 IF a$="**" THEN  RETURN
1080 LET h=CODE a$(1)-48
1085 LET h=h-7*(a$(1)>"@")
1090 LET l=CODE a$(2)-48
1095 LET l=l-7*(a$(2)>"@")
1100 POKE a,16*h+l: LET a=a+1
1105 GO TO 1070
1110 REM
1115 REM  DATI PROGRAMMA L.M.
1120 REM
1125 REM        SUB SHBSX
1130 REM
1135 DATA "01","FF","AF","C5"
1140 DATA "CD","AA","22","E5"
1145 DATA "06","20","A7","CB"
1150 DATA "16","2B","10","FB"
1155 DATA "E1","30","02","CB"
1160 DATA "C6","C1","05","78"
1165 DATA "FE","A7","20","E7"
1170 DATA "C9","**"
```

ASS.SHBSX (SHIFT A SINISTRA)

Per questa routine valgono le stesse considerazioni fatte per ASS.SHBDX.

Lascio al lettore l'esame del listato assembler che non differisce molto da quello precedente. Cambiano infatti qualche istruzione e il modo di operare sulle locazioni video:

- alla variabile XYCO, quella che contiene i valori Y e X del punto della locazione desiderata, bisogna attribuire il valore desiderato in modo differente.

Mentre nello spostamento a destra si operava per locazioni crescenti, cioè partendo dalla locazione 4000 alla 401F (esadecimale), nello spostamento a sinistra si opera in modo inverso. Ciò è stato fatto per semplicità di programmazione.

```
7D00              0000          ORG   32000
                  0010 ;
                  0020 ;              "Routine"
                  0030 ;              "Sistema"
                  0040 ;
22AA              0050 BITC     EQU   22AAH
                  0060 ;
                  0070 ;              "Variab."
                  0080 ;
AFFF              0090 YXCO     EQU   0AFFFH
0020              0100 BYTE     EQU   32
00A7              0110 FINE     EQU   0A7H
                  0120 ;
7D00 01FFAF       0130 SHBSX    LD    BC,YXCO
7D03 C5           0140 LB2      PUSH  BC
7D04 CDAA22       0160          CALL  BITC
7D07 E5           0170          PUSH  HL
7D08 0620         0180          LD    B,BYTE
7D0A A7           0190          AND   A
7D0B CB16         0200 LB3      RL    (HL)
7D0D 2B           0210          DEC   HL
7D0E 10FB         0220          DJNZ  LB3
7D10 E1           0230          POP   HL
7D11 3002         0240          JR    NC,LB4
7D13 CBC6         0250          SET   0,(HL)
7D15 C1           0260 LB4      POP   BC
7D16 05           0270          DEC   B
7D17 78           0280          LD    A,B
7D18 FEA7         0290          CP    FINE
```

```
7D1A 20E7        0300        JR    NZ,LB2
7D1C C9          0310        RET
                 0320        END
LB4     7D15
LB3     7D0B
LB2     7D03
SHBSX   7D00
FINE    00A7
BYTE    0020
YXCO    AFFF
BITC    22AA
#       6062
```

ROUTINE DI SPOSTAMENTO VERTICALE

Le due routine che di seguito vengono descritte non sono altro che l'ovvia espansione delle precedenti. Realizzano lo spostamento verticale nelle due direzioni. Si usa anche qui la subroutine BITC che abbiamo già visto. La routine SHBVS sposta il contenuto del video di un pixel verso l'alto, mentre con la routine SHBVG, si ottiene l'effetto opposto.

SHBVS

Questo programma sposta verso l'alto la prima riga dello schermo. La routine è leggermente più lunga delle precedenti poichè è relativamente più complicato gestire lo spostamentó verso l'alto o verso il basso dello schermo.

```
1000 REM
1005 REM ROUTINE DI SHIFT
1010 REM   VERSO L'ALTO
1015 REM
```

```
1020 CLEAR 31999: GO SUB 1065
1025 PRINT "Questa e' la";
1030 PRINT " routine di SHIFT ";
1035 PRINT "VS"
1040 RANDOMIZE USR 32000
1045 PAUSE 1: GO TO 1040
1050 REM
1055 REM  HEX-CODE Caricatore
1060 REM
1065 LET a=32000
1070 READ a$
1075 IF a$="**" THEN  RETURN
1080 LET h=CODE a$(1)-48
1085 LET h=h-7*(a$(1)>"@")
1090 LET l=CODE a$(2)-48
1095 LET l=l-7*(a$(2)>"@")
1100 POKE a,16*h+l: LET a=a+1
1105 GO TO 1070
1110 REM
1115 REM  DATI PROGRAMMA L.M.
1120 REM
1125 REM        SUB SHBVS
1130 REM
1135 DATA "01","00","AF","3E"
1140 DATA "20","08","C5","CD"
1145 DATA "AA","22","C1","C5"
1150 DATA "7E","F5","EB","C5"
1155 DATA "05","CD","AA","22"
1160 DATA "C1","7E","12","05"
1165 DATA "78","FE","A7","20"
1170 DATA "F1","F1","77","C1"
1175 DATA "79","C6","08","4F"
1180 DATA "08","3D","20","DD"
1185 DATA "C9","**"
```

ASS.SHBVS (SHIFT VERSO l'ALTO)

Commento al listato:

LINEE da 0130 a 0210:

- si carica BC con le modalità viste in precedenza e il registro A con il numero di byte sulla riga da elaborare e si salva sia questo valore nella coppia di registri ausiliari A'F', sia BC sullo stack. Si chiama la routine che calcola l'indirizzo della locazione video da elaborare per prima, se ne carica il valore in A e lo si salva sullo stack. Ciò viene fatto perchè nello spostamento tale valore andrebbe perso.

LINEE da 0220 a 0320:

- ciclo di trasferimento tra il contenuto della linea sottostante a quella sopra. Per linee si intende ovviamente linee dello spessore di un pixel (puntino sullo schermo).

LINEE da 0330 a 0420:

- bisogna premettere una spiegazione prima di proseguire: lo spostamento avviene per colonne ed è esteso fino a FINE linee ed a BYTE colonne.

Ecco che dopo aver prelevato dallo stack il valore di A salvato in precedenza e memorizzato al posto giusto (come sempre nulla della pagina grafica va perso), per ottenere l'indirizzo della prossima colonna bisogna riprendere le coordinate in BC dallo stack ed aggiungere a C, che è l'ascissa, il valore 8 che è lo scarto tra una colonna e l'altra e all'occorrenza proseguire il ciclo.

```
7D00             0000           ORG   32000
                 0010 ;
                 0020 ;               "Routine"
                 0030 ;               "Sistema"
                 0040 ;
22AA             0050 BITC      EQU   22AAH
                 0060 ;
                 0070 ;               "Variab."
                 0080 ;
AF00             0090 YXCO      EQU   0AF00H
0020             0100 BYTE      EQU   20H
00A7             0110 FINE      EQU   0A7H
                 0120 ;
7D00 0100AF      0130 SHBVS LD        BC,YXCO
7D03 3E20        0140           LD    A,BYTE
7D05 08          0150 LB12      EX    AF,A'F'
7D06 C5          0160           PUSH  BC
```

```
7D07 CDAA22    0170          CALL  BITC
7D0A C1        0180          POP   BC
7D0B C5        0190          PUSH  BC
7D0C 7E        0200          LD    A,(HL)
7D0D F5        0210          PUSH  AF
7D0E EB        0220 EXX      EX    DE,HL
7D0F C5        0230          PUSH  BC
7D10 05        0240          DEC   B
7D11 CDAA22    0250          CALL  BITC
7D14 C1        0260          POP   BC
7D15 7E        0270          LD    A,(HL)
7D16 12        0280          LD    (DE),A
7D17 05        0290          DEC   B
7D18 78        0300          LD    A,B
7D19 FEA7      0310          CP    FINE
7D1B 20F1      0320          JR    NZ,EXX
7D1D F1        0330          POP   AF
7D1E 77        0340          LD    (HL),A
7D1F C1        0350          POP   BC
7D20 79        0360          LD    A,C
7D21 C608      0370          ADD   8
7D23 4F        0380          LD    C,A
7D24 08        0390          EX    AF,A'F'
7D25 3D        0400          DEC   A
7D26 20DD      0410          JR    NZ,LB12
7D28 C9        0420          RET
               0430          END

EXX     7D0E
LB12    7D05
SHBVS   7D00
FINE    00A7
BYTE    0020
YXCO    AF00
BITC    22AA
#       6062
```

SHBVG

Ovvio complemento al precedente programma, ecco come avviene lo spostamento verso il basso della solita riga.

Come per gli spostamenti orizzontali è possibile modificare il valore di alcuni byte in L/M per estendere lo spostamento a porzioni ben precise dello schermo.

```
1000 REM
1005 REM ROUTINE DI SHIFT
1010 REM  VERSO IL BASSO
1015 REM
1020 CLEAR 31999: GO SUB 1065
1025 PRINT "Questa e' la";
1030 PRINT " routine di SHIFT ";
1035 PRINT "VG"
1040 RANDOMIZE USR 32000
1045 PAUSE 1: GO TO 1040
1050 REM
1055 REM  HEX-CODE Caricatore
1060 REM
1065 LET a=32000
1070 READ a$
1075 IF a$="**" THEN  RETURN
1080 LET h=CODE a$(1)-48
1085 LET h=h-7*(a$(1)>"@")
1090 LET l=CODE a$(2)-48
1095 LET l=l-7*(a$(2)>"@")
1100 POKE a,16*h+l: LET a=a+1
1105 GO TO 1070
1110 REM
1115 REM  DATI PROGRAMMA L.M.
1120 REM
1125 REM        SUB SHBVG
1130 REM
1135 DATA "01","00","A8","3E"
1140 DATA "20","08","C5","CD"
1145 DATA "AA","22","C1","C5"
1150 DATA "7E","F5","EB","C5"
1155 DATA "04","CD","AA","22"
1160 DATA "C1","7E","12","04"
1165 DATA "78","FE","AF","20"
1170 DATA "F1","F1","77","C1"
1175 DATA "79","C6","08","4F"
1180 DATA "08","3D","20","DD"
1185 DATA "C9","**"
```

ASS.SHBVG (SHIFT VERSO IL BASSO)

Anche in questo caso la routine non differisce molto da quella precedente. Bisogna fare attenzione a ben inizializzare le 2 variabili YXCO e FINE il cui valore viene attribuito in maniera inversa a quanto fatto nella routine precedente.

```
7D00            0000          ORG   32000
                0010 ;
                0020 ;              "Routine"
                0030 ;              "Sistema"
                0040 ;
22AA            0050 BITC     EQU   22AAH
                0060 ;
                0070 ;              "Variab."
                0080 ;
A800            0090 YXCO     EQU   0A800H
0020            0100 BYTE     EQU   20H
00AF            0110 FINE     EQU   0AFH
                0120 ;
7D00 0100A8     0130 SHBVG    LD    BC,YXCO
7D03 3E20       0140          LD    A,BYTE
7D05 08         0150 LB12     EX    AF,A'F'
7D06 C5         0160          PUSH  BC
7D07 CDAA22     0170          CALL  BITC
7D0A C1         0180          POP   BC
7D0B C5         0190          PUSH  BC
7D0C 7E         0200          LD    A,(HL)
7D0D F5         0210          PUSH  AF
7D0E EB         0220 EXX      EX    DE,HL
7D0F C5         0230          PUSH  BC
7D10 04         0240          INC   B
7D11 CDAA22     0250          CALL  BITC
7D14 C1         0260          POP   BC
7D15 7E         0270          LD    A,(HL)
7D16 12         0280          LD    (DE),A
7D17 04         0290          INC   B
7D18 78         0300          LD    A,B
7D19 FEAF       0310          CP    FINE
7D1B 20F1       0320          JR    NZ,EXX
```

```
7D1D F1        0330      POP  AF
7D1E 77        0340      LD   (HL),A
7D1F C1        0350      POP  BC
7D20 79        0360      LD   A,C
7D21 C608      0370      ADD  8
7D23 4F        0380      LD   C,A
7D24 08        0390      EX   AF,A'F'
7D25 3D        0400      DEC  A
7D26 20DD      0410      JR   NZ,LB12
7D28 C9        0420      RET
               0430      END

EXX     7D0E
LB12    7D05
SHBVG   7D00
FINE    00AF
BYTE    0020
YXCO    A800
BITC    22AA
#       6062
```

SUPER CHR$

È un programma molto utile che serve per ingrandire un messaggio su video da 2 fino 7 volte (o più). Il suo funzionamento è semplice. Il messaggio che si vuole stampare viene scritto in una zona dello schermo quindi la matrice di punti (8×8) di ciascun carattere è scandita dall'istruzione POINT per determinare se un punto sul video è acceso o spento. Se acceso bisognerà ingrandirlo secondo il parametro I da noi introdotto.

La stringa da ingrandire è stampata in alto a sinistra mentre l'ingrandimento è stampato poco più sotto. L'ingrandimento possibile è comunque superiore a 7. Maggiore è l'ingrandimento, minore è ovviamente il numero di caratteri su video. Il programma si può modificare a piacere e si può accodare ad un programma principale come subroutine essendo abbastanza veloce nell'elaborazione.

Un consiglio: si può evitare di stampare in alto a sinistra il messaggio da ingrandire, che potrebbe alterare il display in quella posizione, utilizzando

una tecnica altrove usata (vedi CHR$ EDITOR) che consiste nell'elaborare i dati che definiscono il carattere prelevandoli dalla RAM (U.D.G.) o ROM (A.S.C.I.I.).

```
1000 BORDER 1: INK 7: PAPER 1
1005 OVER 0: BRIGHT 0: FLASH 0
1010 CLS
1015 PRINT AT 1,9; INK 6; PAPER 2; FLASH
 1; INVERSE 1;"SUPER CARACTHERS"
1020 PRINT : PRINT ;"QUESTO PROGRAMMA PE
RMETTE DI VI-"
1025 PRINT "SUALIZZARE MESSAGGI INGRANDI
TI"
1030 PRINT "DA 2 A 7 VOLTE."
1035 PRINT : PRINT "IL NUMERO DEI CARATT
ERI E':": PRINT
1040 PRINT "16 PER INGRANDIMENTO = 2"
1045 PRINT "10 PER INGRANDIMENTO = 3"
1050 PRINT "08 PER INGRANDIMENTO = 4"
1055 PRINT "06 PER INGRANDIMENTO = 5"
1060 PRINT "05 PER INGRANDIMENTO = 6"
1065 PRINT "04 PER INGRANDIMENTO = 7"
1070 PRINT AT 21,3;"BATTI <ENTER> PER IN
IZIARE "
1075 PAUSE 0: CLS
1080 INPUT "MESSAGGIO:";M$
1085 LET CH=LEN M$
1090 INPUT "INGRANDIMENTO (2-7):";I: CLS
1095 PRINT  FLASH 1;AT 0,0;M$
1100 FOR X=0 TO CH*8-1
1105 FOR Y=175 TO 168 STEP -1
1110 LET Z=175-Y
1115 IF POINT (X,Y) THEN  GO SUB 1155
1120 NEXT Y: NEXT X
1125 FOR J=0 TO 31
1130 PRINT AT 0,J;" ";
1135 NEXT J
1140 INPUT "ALTRO MESSAGGIO ?:";E$
1145 IF E$="S" OR E$="s" THEN  GO TO 108
0
1150 STOP
1155 FOR V=0 TO I-1
```

```
1160 PLOT I*X+V,165-Z*I
1165 DRAW 0,I-1: NEXT V
1170 RETURN
```

SUPERCHR$

OTTOGRAF

Questo programmino è il naturale complemento a SUPER CHR$ prima descritto.

La sua funzione è di permettere la scrittura di un messaggio in 8 direzioni sullo schermo. Dapprima si stampa il messaggio in zona nota del video, poi l'istruzione POINT fa un test sulla matrice di pixel di ciascun carattere che si vuol stampare e si disegna il pixel corrispondente, a partire dalle coordinate stabilite, secondo la direzione voluta.

```
1000 INPUT "Direzione ? ";d$
1005 LET a=VAL d$
1010 INPUT "Messaggio ? ";m$
1015 REM
1020 INPUT "X= ? ";x
1025 INPUT "Y= ? ";y
1030 PRINT AT 21,0;m$
1035 FOR n=0 TO LEN m$*8-1
1040 FOR o=0 TO 7
1045 IF POINT (n,o) THEN  GO SUB 1065+5*
a
1050 NEXT o: NEXT n
1055 GO TO 1000
1060 REM
1065 PLOT o-x,n+y: RETURN
1070 PLOT n+x-o,n+y+o: RETURN
1075 PLOT n+x,o+y: RETURN
1080 PLOT n+x+o,o+y-n: RETURN
1085 PLOT o+x,n-y: RETURN
1090 PLOT n-x-o,n-y+o: RETURN
1095 PLOT n-x,o-y: RETURN
1100 PLOT n-x+o,o-y-n: RETURN
```

PAINT

Questo programma è un esempio di come utilizzare l'omonima routine in L/M di cui, come al solito, si da più avanti il listato assembler.

Dopo aver caricato il linguaggio macchina all'indirizzo 32000, il programma mostra un grafico a barre generato casualmente.

È proprio in applicazioni di questo tipo che la potenza della routine di

PAINT (to paint in inglese significa verniciare, colorare...) si fa vedere.

Infatti risulta particolarmente adatta a riempire di colore zone dello schermo definite da figure geometriche molto regolari come i rettangoli in questo caso.

Le figure contorte e molto intersecate sono più difficili da riempire e per ognuna di esse bisognerà prendere più punti, scelti in modo strategico, ove indirizzare la routine PAINT.

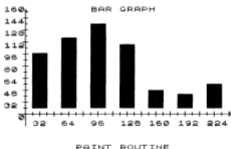
Usare quest'ultima è molto semplice. Si deve dare il seguente comando:

RANDOMIZE x+y*USR 32000

oppure

LET a=x+y*USR 32000

X e Y sono due variabili cioè l'ascissa e l'ordinata del punto ove si vuole cominci la colorazione della figura.

PAINT ROUTINE

```
1000 REM
1005 REM          PAINT
1010 REM         ESEMPIO
1015 REM ISTOGRAMMA A BARRE
1020 REM
1025 CLEAR 31999: GO SUB 1190
1030 CLS : DIM y(7): OVER 0
1035 LET p=32000: FOR x=1 TO 7
```

```
1040 LET y(x)=10+110*RND
1045 PLOT 32*x,40: DRAW 0,y(x)
1050 DRAW 16,0: DRAW 0,-y(x)
1055 DRAW -16,0: NEXT x
1060 PLOT 16,32: DRAW 235,0
1065 DRAW -3,3: DRAW 0,-6
1070 DRAW 3,3: PLOT 24,16
1075 DRAW 0,155: DRAW -3,-3
1080 DRAW 6,0: DRAW -3,3
1085 FOR x=32 TO 240 STEP 16
1090 PLOT x,34: DRAW 0,-4
1095 PLOT x+8,35: DRAW 0,-6
1100 NEXT x
1105 PRINT AT 0,12;"BAR GRAPH"
1110 FOR y=40 TO 160 STEP 16
1115 PLOT 26,y: DRAW -4,0
1120 PLOT 27,y+8: DRAW -6,0
1125 NEXT y
1130 PRINT AT 18,2;0
1135 FOR y=32 TO 160 STEP 16
1140 PRINT AT 20-y/8,0;y
1145 NEXT y
1150 FOR x=32 TO 240 STEP 32
1155 PRINT AT 19,x/8;x
1160 NEXT x
1165 FOR x=1 TO 7: INK 7-x
1170 LET y=(8+32*x)+45*USR p
1175 NEXT x
1180 PRINT #0;TAB 10;"PAINT ROUTINE"
1185 PAUSE 1: PAUSE 0: GO TO 1030
1190 REM
1195 REM  HEX-CODE Caricatore
1200 REM
1205 LET a=32000
1210 PRINT AT 10,11;"ATTENDERE"
1215 BEEP .01,65: READ a$
1220 IF a$="**" THEN  RETURN
1225 LET h=CODE a$(1)-48
1230 LET h=h-7*(a$(1)>"@")
1235 LET l=CODE a$(2)-48
1240 LET l=l-7*(a$(2)>"@")
1245 POKE a,16*h+l: LET a=a+1
1250 GO TO 1215
```

```
1255 REM
1260 REM      DATI PROGRAMMA L.M.
1265 REM
1270 REM            PAINT
1275 REM
1280 DATA "2A","65","5C","ED","5B","63"
1285 DATA "5C","A7","ED","52","7D","FE"
1290 DATA "0A","DA","8B","28","CD","94"
1295 DATA "1E","F5","CD","94","1E","5F"
1300 DATA "F1","57","21","01","01","42"
1305 DATA "4B","E5","D5","C5","CD","E5"
1310 DATA "22","C1","D1","E1","CD","54"
1315 DATA "1F","30","45","7D","81","4F"
1320 DATA "28","13","E5","D5","C5","CD"
1325 DATA "AA","22","47","04","7E","07"
1330 DATA "10","FD","E6","01","C1","D1"
1335 DATA "E1","28","DA","AF","95","6F"
1340 DATA "FE","01","20","D2","7C","80"
1345 DATA "47","28","1A","FE","AF","30"
1350 DATA "16","2E","01","4B","E5","D5"
1355 DATA "C5","CD","AA","22","47","04"
1360 DATA "7E","07","10","FD","E6","01"
1365 DATA "C1","D1","E1","28","B3","AF"
1370 DATA "94","67","FE","01","20","AB"
1375 DATA "CD","28","2D","C3","28","2D"
1380 DATA "**"
```

ASS.PAINT

Quanto segue è il commento della routine PAINT. Questa ha la funzione di riempire zone dello schermo attorno ad un punto specificato. Ad esempio scegliendo un punto all'interno di un rettangolo, e chiamando questa routine, il risultato sarà quello di riempire la zona racchiusa dal perimetro della figura stessa.

Cioè si disegnano tutti i punti attorno a quello iniziale secondo lo schema seguente:

– il riempimento avviene secondo linee orizzontali a partire dal punto specificato incrementandone l'ascissa fino ad incontrare un punto già disegnato.

In tal caso la routine torna al punto di partenza invertendo il senso di marcia, anche qui procedendo alla ricerca di un punto già presente. Occorre successivamente passare alla linea superiore e allo scopo si incrementa l'ordinata iniziale. Il procedimento è uguale anche per tutti i punti che stanno al di sotto di quello dato e comunque racchiusi dal perimetro della figura.

La routine non è il massimo dell'intelligenza nel senso che se la figura è molto complessa, contorta o intersecata da altre figure, il riempimento avviene in modo non completo.

A questo apparente difetto si può ovviare scegliendo dei punti adeguati all'interno della figura. Per riempire con un colore preciso bisogna, prima di chiamare la routine, dare il comando INK C, dove C è il colore che si desidera. Nell'applicazione Basic di PAINT ciò è chiaramente visibile.

Commento:

LINEE da 170 a 230:

– in queste linee viene fatto un test per controllare se le coordinate grafiche del punto sono state passate correttamente al L/M. In caso contrario si da un messaggio di errore saltando alla subroutine 288BH.

LINEE da 240 a 320:

– tali coordinate vengono a turno prelevate dallo Stack ove il BASIC memorizza il valore delle variabili attualmente in uso e sono poste nei registri DE e BC.

LINEE da 330 a 450:

– si disegna il punto di coordinate BC e si controlla la pressione del tasto BREAK.

LINEE da 460 a 590:

– questa serie di istruzioni equivalgono all'istruzione POINT in BASIC. In base al risultato di questa operazione si decide se cambiare il senso di «plottaggio» orizzontale del punto (da destra verso sinistra) o proseguire nella stessa direzione fino ad incontrare un punto già disegnato.

LINEE da 600 a 860:

— se tale cambiamento di direzione è già avvenuto, si provvede a cambiare la direzione verticale passando alla linea di pixel immediatamente superiore.

LINEE da 870 a 930:

— il tutto viene ripetuto per la porzione di schermo al di sotto del punto iniziale ove si è iniziata l'elaborazione.

LINEE da 920 a 930:

— se ciò non fosse possibile vengono ripristinate le locazioni di RAM dalle quali si sono prelevate le coordinate del punto saltando per due volte di seguito alla subroutine 2D28H. Vengono memorizzati due numeri a caso, ma ciò è necessario prima di ritornare al BASIC.

```
7D00              0000        ORG   32000
                  0010 ;
                  0020 ;            "Routine"
                  0030 ;            "Sistema"
                  0040 ;
22E5              0050 PLOT   EQU   22E5H
22AA              0060 BITC   EQU   22AAH
1E94              0070 LDAC   EQU   1E94H
2D28              0080 STACK  EQU   2D28H
288B              0090 ERR    EQU   288BH
1F54              0100 BREAK  EQU   1F54H
                  0110 ;
                  0120 ;            "Variab."
                  0130 ;
5C65              0140 STED   EQU   5C65H
5C63              0150 STBT   EQU   5C63H
                  0160 ;
7D00 2A655C       0170 INIT   LD    HL,(STED)
7D03 ED5B635C     0180        LD    DE,(STBT)
7D07 A7           0190        AND   A
7D08 ED52         0200        SBC   HL,DE
7D0A 7D           0210        LD    A,L
7D0B FE0A         0220        CP    10
7D0D DA8B28       0230        JP    C,ERR
```

```
7D10 CD941E    0240 LDAA   CALL LDAC
7D13 F5        0250        PUSH AF
7D14 CD941E    0260        CALL LDAC
7D17 5F        0270        LD   E,A
7D18 F1        0280        POP  AF
7D19 57        0290        LD   D,A
7D1A 210101    0300 PAINT  LD   HL,0101H
7D1D 42        0310 UP     LD   B,D
7D1E 4B        0320 NXTR   LD   C,E
7D1F E5        0330 NXTP   PUSH HL
7D20 D5        0340        PUSH DE
7D21 C5        0350        PUSH BC
7D22 CDE522    0360        CALL PLOT
7D25 C1        0370        POP  BC
7D26 D1        0380        POP  DE
7D27 E1        0390        POP  HL
7D28 CD541F    0400        CALL BREAK
7D2B 3045      0410        JR   NC,EXIT
7D2D 7D        0420        LD   A,L
7D2E 81        0430        ADD  C
7D2F 4F        0440        LD   C,A
7D30 2813      0450        JR   Z,INV
7D32 E5        0460        PUSH HL
7D33 D5        0470        PUSH DE
7D34 C5        0480        PUSH BC
7D35 CDAA22    0490        CALL BITC
7D38 47        0500        LD   B,A
7D39 04        0510        INC  B
7D3A 7E        0520        LD   A,(HL)
7D3B 07        0530 P1     RLCA
7D3C 10FD      0540        DJNZ P1
7D3E E601      0550        AND  1
7D40 C1        0560        POP  BC
7D41 D1        0570        POP  DE
7D42 E1        0580        POP  HL
7D43 28DA      0590        JR   Z,NXTP
7D45 AF        0600 INV    XOR  A
7D46 95        0610        SUB  L
7D47 6F        0620        LD   L,A
7D48 FE01      0630        CP   1
7D4A 20D2      0640        JR   NZ,NXTR
7D4C 7C        0650        LD   A,H
7D4D 80        0660        ADD  B
```

```
7D4E 47       0670          LD   B,A
7D4F 281A     0680          JR   Z,NXTC
7D51 FEAF     0690          CP   175
7D53 3016     0700          JR   NC,NXTC
7D55 2E01     0710          LD   L,1
7D57 4B       0720          LD   C,E
7D58 E5       0730          PUSH HL
7D59 D5       0740          PUSH DE
7D5A C5       0750          PUSH BC
7D5B CDAA22   0760          CALL BITC
7D5E 47       0770          LD   B,A
7D5F 04       0780          INC  B
7D60 7E       0790          LD   A,(HL)
7D61 07       0800 P2       RLCA
7D62 10FD     0810          DJNZ P2
7D64 E601     0820          AND  1
7D66 C1       0830          POP  BC
7D67 D1       0840          POP  DE
7D68 E1       0850          POP  HL
7D69 28B3     0860          JR   Z,NXTR
7D6B AF       0870 NXTC     XOR  A
7D6C 94       0880          SUB  H
7D6D 67       0890          LD   H,A
7D6E FE01     0900          CP   1
7D70 20AB     0910          JR   NZ,UP
7D72 CD282D   0920 EXIT     CALL STACK
7D75 C3282D   0930          JP   STACK
              0940          END
EXIT    7D72
NXTC    7D6B
P2      7D61
INV     7D45
P1      7D3B
NXTP    7D1F
NXTR    7D1E
UP      7D1D
PAINT   7D1A
LDAA    7D10
INIT    7D00
STBT    5C63
STED    5C65
BREAK   1F54
ERR     288B
```

```
STACK  2D28
LDAC   1E94
BITC   22AA
PLOT   22E5
#      6062
```

CHR$ EDITOR

Questo character editor è molto differente da tutti quelli che ho fino ad ora provato e visto. Intanto, come al solito in questi programmi, ci si basa su una routine in L/M di ben 382 bytes per ottenere dei piacevoli e utili effetti, altrimenti non ottenibili.

Come dice il nome, il programma serve per creare, modificare, in una sola parola, editare, quei caratteri che utilizzeremo nelle nostre applicazioni grafiche.

Usualmente questi caratteri verranno memorizzati nell'area degli U.D.G. ma, modificando opportunamente il programma, si possono memorizzare in altre zone di RAM.

Lo schermo è stato diviso in due parti e si è scelto un metodo originale per disegnare i caratteri. Questi infatti si disegnano nella parte superiore dello schermo in un formato che è otto volte pii grande di quello originale visibile nella parte in basso a sinistra.

In altre parole:

si ha a disposizione una griglia di ben 32 per 32 caselle su cui disegnare.

Di queste solo 32 × 16 sono contemporaneamente visibili sullo schermo e corrispondono alle prime 16 righe per 32 colonne del video.

Come vedere le rimanenti caselle?

Occorre dare qualche spiegazione.

Nella parte superiore del video abbiamo un cursore che si muove con le modalità che più avanti verranno discusse.

Scrivere un punto (che è poi una casella) alla posizione attuale del cursore è molto semplice: o si disegna il carattere che possiamo vedere sopra il

tasto con il numero 8 o si elabora il byte degli attributi che appartiene a quella casella.

È stata scelta quest'ultima possibilità e ciò per ragioni molto pratiche.

La parte superiore dello schermo si comporta come una finestra di 512 caselle e per mostrare le rimanenti 512 si fa ricorso a diverse routine in L/M il cui compito principale è di spostare il contenuto delle prime 512 locazioni della zona degli attributi del video in una zona ben precisa della RAM.

Ciò viene fatto in generale modificando in più o in meno un puntatore che agisce su questa zona di RAM, in modo tale da realizzare una serie di «scrolling» nelle due direzioni verticali qualora i dati vengano poi ritrasferiti al video.

Per concludere è possibile editare 16 caratteri grafici contemporaneamente, costruendoli su una matrice 8 volte più grande di quella originale.

In ogni momento è possibile osservare le dimensioni reali di ciò che si sta editando facendo riferimento in basso a sinistra nel riquadro indicato dalla freccia lampeggiante.

Questo riquadro è la copia esatta di ciò che esiste nella parte superiore dello schermo e ne subisce ogni modifica, ad esempio lo spostamento nelle quattro direzioni premendo i seguenti tasti:

```
   W
A     S
   Z
```

Per spostare il cursore si usano i soliti tasti

5 6 7 8

da soli o in contemporanea con il tasto CAPS SHIFT o SYMBOL SHIFT.

In tal caso si disegna un punto o lo si cancella. Per rapide cancellazioni si usa il tasto 0.

Premendo invece i tasti

```
   I
K     L
   M
```

si sposta il cursore di otto posizioni nelle quattro rispettive direzioni.

Premendo il tasto H si porta il cursore in alto a sinistra, mentre premendo il tasto B si porta il cursore in basso a sinistra. Il tasto C esegue una copia su stampante di ciò che appare su video.

Il tasto P permette di prendere un carattere e di disegnarlo alla posizione del cursore. Non c'è restrizione sul tipo di carattere: può essere grafico, compresi quelli su ogni tasto numerico, o qualsiasi altro presente sulla tastiera.

Ogni carattere può inoltre essere disegnato in reverse e in modo speculare rispetto all'originale.

Il tasto O permette la memorizzazione in uno degli U.D.G., del carattere la cui matrice otto per otto è dedotta dalla posizione attuale del cursore.

Dopo aver digitato questo programma bisogna salvarlo su nastro con un SAVE"CHR$ EDITOR"LINE 2090, accendere e spegnere il computer, e digitare il programma che contiene il L/M (VIDEO TRASF.) il cui prodotto andrà salvato in coda al programma precedente.

```
1000 REM
1005 REM  CHR$ EDITOR
1010 REM
1015 LET ink=7: LET pap=1
1020 BORDER 1: INK ink: PAPER pap
1025 GO SUB 1435: REM INIZIO
1030 GO SUB 1845: REM CRS.ON
1035 PAUSE 2
1040 GO SUB 1870: REM CRS.OFF
```

```
1045 LET k$=INKEY$: LET k=CODE k$
1050 IF k$="" THEN  GO TO 1030
1055 REM
1060 IF k=08 THEN  GO SUB 1775: GO TO 1030
1065 IF k=10 THEN  GO SUB 1780: GO TO 1030
1070 IF k=11 THEN  GO SUB 1785: GO TO 1030
1075 IF k=09 THEN  GO SUB 1790: GO TO 1030
1080 REM
1085 IF k$="5" THEN  GO SUB 1720: GO TO 1030
1090 IF k$="6" THEN  GO SUB 1725: GO TO 1030
1095 IF k$="7" THEN  GO SUB 1740: GO TO 1030
1100 IF k$="8" THEN  GO SUB 1755: GO TO 1030
1105 REM
1110 IF k$="%" THEN  GO SUB 1810: GO TO 1030
1115 IF k$="&" THEN  GO SUB 1815: GO TO 1030
1120 IF k$="'" THEN  GO SUB 1820: GO TO 1030
1125 IF k$="(" THEN  GO SUB 1825: GO TO 1030
1130 IF k$="0" THEN  GO SUB 1810: GO TO 1030
1135 REM
1140 IF k$="A" THEN  GO SUB 1630: GO TO 1030
1145 IF k$="S" THEN  GO SUB 1635: GO TO 1030
1150 IF k$="W" THEN  GO SUB 1640: GO TO 1030
1155 IF k$="Z" THEN  GO SUB 1645: GO TO 1030
1160 REM
1165 IF k$="I" THEN  GO SUB 1920: GO TO 1030
1170 IF k$="M" THEN  GO SUB 1930: GO TO 1030
1175 IF k$="K" THEN  GO SUB 1950: GO TO 1030
1180 IF k$="J" THEN  GO SUB 1940: GO TO 1030
1185 REM
1190 IF k$="P" THEN  GO SUB 1305
1195 IF k$="O" THEN  GO SUB 1235
1200 IF k$="H" THEN  GO SUB 1910
1205 IF k$="B" THEN  GO SUB 1915
1210 IF k$="C" THEN  GO SUB 1975
1215 GO TO 1030
1220 REM
1225 REM  SUB.STORE U.D.G.
1230 REM
1235 POKE 23617,2
1240 INPUT  FLASH u;"A-U scegli....";a$
1245 IF a$>CHR$ 164 THEN  GO TO 1235
1250 IF a$<CHR$ 144 THEN  GO TO 1400
1255 LET j=INT (x/o)
```

```
1260 LET j=j+q+32*(3-INT ((m-b)/o))
1265 FOR i=z TO 7
1270 POKE (USR a$+i),PEEK (j+256*i): NEXT i
1275 OVER z: FOR i=z TO 20
1280 PRINT AT 17,6+i;CHR$ (144+i)
1285 NEXT i: OVER u: GO TO 1400
1290 REM
1295 REM  SUB. PICK CHR$
1300 REM
1305 INPUT "Carattere riflesso ? ";a$
1310 IF a$="S" THEN  POKE gr+15,31
1315 INPUT  INVERSE u;"Carattere in inverse
? ";a$
1320 LET f=z: IF a$="S" THEN  LET f=u
1325 POKE 23617,2
1330 INPUT  FLASH u;"Scegli il carattere..";
a$
1335 IF a$>CHR$ 164 THEN  GO TO 1305
1340 IF a$="" THEN  GO TO 1400
1345 IF a$<CHR$ 128 THEN  LET i=PEEK 23606+2
56*PEEK 23607+o*CODE a$: GO TO 1360
1350 IF a$<CHR$ 144 THEN  LET i=mem+1: POKE
gr+1,CODE a$: RANDOMIZE USR gr: GO TO 1360
1355 LET i=USR a$
1360 LET x1=x
1365 FOR j=i TO i+7: LET x=x1
1370 POKE mem,PEEK j
1375 FOR v=u TO o
1380 IF USR rla-f THEN  GO SUB 1790: GO TO 1
390
1385 GO SUB 1825
1390 NEXT v: GO SUB 1725: NEXT j
1395 LET x=x1: POKE gr+15,23
1400 POKE 23617,z: POKE 23658,o
1405 PRINT #1;AT 0,0;"U.D.G ---> O=MEMORIZZA
  P=PRENDE"
1410 PRINT #0;"H)OME  B)OTTOM   C)OPY  0=DEL
ETE"
1415 RETURN
1420 REM
1425 REM  SUB. GRIGLIA
1430 REM
1435 LET z=0: LET u=1: LET o=8
```

```
1440 LET q=20545: OVER u: CLS
1445 LET m=31: LET t=255: LET p=175
1450 LET l=128: LET c=15: LET a=o*(21-c)
1455 LET b=p-a: LET mem=23296
1460 LET gr=31359: LET rla=gr+9
1465 LET new=31000: LET plus=31014
1470 LET minus=31035: LET dx=31081
1475 LET sx=31140: LET su=31199
1480 LET giu=31276: LET ramv=31060
1485 LET vram=31068
1490 FOR y=a TO p STEP o
1495 PLOT z,y: DRAW t,z: NEXT y
1500 FOR x=z TO t STEP o
1505 PLOT x,a: DRAW z,b: NEXT x
1510 PLOT z,p: DRAW t,z: DRAW z,-b
1515 FOR y=7 TO m STEP o
1520 PLOT z,y: DRAW 6,z: NEXT y
1525 FOR x=16 TO 40 STEP o
1530 PLOT x,33: DRAW z,6: NEXT x
1535 PLOT  FLASH u;6,33: DRAW -4,4
1540 PLOT 6,33: DRAW -2,z: PLOT 6,33: DRAW z
,2
1545 FOR x=z TO 20
1550 PRINT  INVERSE u;AT 18,6+x;CHR$ (65+x)
1555 NEXT x: GO SUB 1275
1560 FOR i=z TO u: FOR j=z TO m STEP o
1565 PRINT AT i*o,j; FLASH u;"+"
1570 NEXT j: NEXT i
1575 PRINT AT 19,6;"CURS 5 6 7 8 + CAPS+SYMB
OL"
1580 PRINT TAB 6;"JUMP J M I K * 8 Posizioni
"
1585 PRINT TAB 6;"SHFT A Z W S / Sposta Fig.
"
1590 PRINT AT 16,28;"CHR$"
1595 PRINT AT 17,28;"EDT."
1600 POKE 23658,o
1605 LET x=z: LET y=x: LET b=x
1610 RANDOMIZE USR new: RETURN
1615 REM
1620 REM    SUB SHIFT SX/DX
1625 REM
1630 RANDOMIZE USR sx: BEEP .1,b: RETURN
```

```
1635 RANDOMIZE USR dx: BEEP .1,-b: RETURN
1640 RANDOMIZE USR su: BEEP .1,y: RETURN
1645 RANDOMIZE USR giu: BEEP .1,-y: RETURN
1650 REM
1655 REM  DISEGNA IL PUNTO
1660 REM
1665 PRINT AT y,x; FLASH (ATTR (y,x)>=l); PA
PER ink;" "
1670 PLOT  OVER z;x+o,m-b: RETURN
1675 REM
1680 REM  CANCELLA IL PUNTO
1685 REM
1690 PRINT AT y,x; FLASH (ATTR (y,x)>=l); PA
PER pap;" "
1695 PLOT  OVER z; INVERSE u;x+o,m-b
1700 RETURN
1705 REM
1710 REM  SET DELLA POSIZIONE
1715 REM
1720 BEEP .1*(x=z),y: LET x=x-(x>z): RETURN
1725 LET b=b+(b<m): BEEP .1*(b=m),-y
1730 IF y=c THEN  RANDOMIZE USR plus: BEEP .
1*(b=m),-20
1735 LET y=y+(y<c): RETURN
1740 LET b=b-(b>z): BEEP .1*(b=z),y
1745 IF y=z THEN  RANDOMIZE USR minus: BEEP
.1*(b=z),20
1750 LET y=y-(y>z): RETURN
1755 BEEP .1*(x=m),-y: LET x=x+(x<m): RETURN
1760 REM
1765 REM  PLOT E SET CURSORI
1770 REM
1775 GO SUB 1665: GO TO 1720
1780 GO SUB 1665: GO TO 1725
1785 GO SUB 1665: GO TO 1740
1790 GO SUB 1665: GO TO 1755
1795 REM
1800 REM  DELETE E SET CURSORI
1805 REM
1810 GO SUB 1690: GO TO 1720
1815 GO SUB 1690: GO TO 1725
1820 GO SUB 1690: GO TO 1740
1825 GO SUB 1690: GO TO 1755
```

```
1830 REM
1835 REM  DISEGNA CURSORI
1840 REM
1845 PRINT AT y,x; INK 9; FLASH (ATTR (y,x))
=l); PAPER o;"+"
1850 PLOT x+o,m-b: RETURN
1855 REM
1860 REM  CANCELLA CURSORI
1865 REM
1870 PRINT AT y,x; FLASH (ATTR (y,x)>=l); PA
PER o;"+"
1875 PLOT  INVERSE z;x+o,m-b
1880 RETURN
1885 REM
1890 REM  SUB HOME / BOTTOM
1895 REM      LEFT & RIGHT
1900 REM        UP/DOWN
1905 REM
1910 LET x=z: FOR i=b TO u STEP -u: GO TO 19
25
1915 LET x=z: FOR i=b TO m: GO TO 1935
1920 FOR i=u TO o
1925 GO SUB 1740: GO TO 1955
1930 FOR i=u TO o
1935 GO SUB 1725: GO TO 1955
1940 FOR i=u TO o
1945 GO SUB 1720: GO TO 1955
1950 FOR i=u TO o: GO SUB 1755
1955 NEXT i: BEEP .1,-y: RETURN
1960 REM
1965 REM  SUB COPY
1970 REM
1975 INPUT  FLASH u;"COPY ?";a$
1980 GO SUB 1400
1985 IF a$<>"S" THEN  RETURN
1990 RANDOMIZE USR vram
1995 LET set=o*ink+ink
2000 LET del=o*pap+ink
2005 FOR i=z TO c: FOR j=z TO m
2010 LET at=ATTR (i,j)
2015 IF at=set OR at=set+l THEN  PRINT AT i,
j; BRIGHT u;"■"
2020 NEXT j: NEXT i: BEEP .1,y
```

```
2025 REM
2030 COPY
2035 REM
2040 FOR i=z TO c: FOR j=z TO m
2045 LET at=ATTR (i,j)
2050 IF at<>del AND at<>del+1 THEN  PRINT AT
 i,j;"█"
2055 NEXT j: NEXT i: BEEP .1,y
2060 RANDOMIZE USR ramv: RETURN
2065 REM
2070 REM  SUB SAVE
2075 REM
2080 CLEAR : SAVE "CHR$ EDITOR" LINE 2090
2085 SAVE "TRASF"CODE 31000,382: STOP
2090 CLEAR 30999: LOAD ""CODE : RUN
```

ASS.VIDEO TRASF

Lo scopo principale di questa serie di routines è di servire il programma CHR$ EDITOR nelle sue funzioni. Tuttavia nessuno vieta di modificarne il contenuto per altri usi.

La funzione principale è quella di muovere su una zona di RAM di 1024 bytes, una finestra virtuale di 512 bytes e mostrarli come attributi delle prime 512 locazioni dello schermo (sul file degli attributi).

In pratica è come se si gestissero gli attributi di 32 linee di schermo anzichè 24.

Ogni elaborazione si basa perciò su un puntatore che opportunamente modificato consente di spaziare su tutta la suddetta zona di RAM, prelevando dal video gli attributi delle prime 16 linee e memorizzandoli in RAM. Modificando poi il puntatore, aggiungendogli non a caso 32, e riportando il tutto di nuovo nelle prime 512 locazioni del file degli attributi, il risultato sarà quello di vedere il tutto spostato verso l'alto. In definitiva si mostrano gli attributi delle linee a partire dalla 2 fino alla 17 delle ipotetiche 32 di cui prima si è parlato.

Il procedimento vale anche se condotto in modo inverso.

Il resto delle routines realizza uno shift circolare nelle quattro direzioni agendo su queste 1024 locazioni di RAM.

Contemporaneamente tale spostamento viene esteso alla zona dello schermo in basso a sinistra.

Commento:

LINEE da 200 a 240:

è il punto in cui si esegue il "set up" delle 1024 locazioni di RAM citate. Esse vengono riempite con il valore delle prime 512 locazioni del file degli attributi (prime 16 linee di schermo per due volte), poi viene memorizzato il valore del puntatore a detta area di RAM.

LINEE da 280 a 380:

queste istruzioni prelevano le solite 512 locazioni e le memorizzano in RAM in base al valore del puntatore, quindi si incrementa quest'ultimo di 32 unità e con il nuovo valore si ritrasferisce tutto al video.

LINEE da 420 a 520:

esattamente come prima ma anzichè aggiungere si sottrae 32 al puntatore.

LINEE da 560 a 680:

tra queste istruzioni vi sono diversi punti di accesso la cui funzione è facilmente intuibile. Le più usate sono quelle di trasferimento dal video alla RAM e viceversa.

LINEE da 720 a 840:

iniziano qui le routines di rotazione e questa chiamata SHDX esegue la rotazione verso destra di tutte le 1024 locazioni già dette.

LINEE DA 880 a 1030:

SHBDX esegue subito dopo la rotazione verso destra, pixel per pixel, della porzione di video a partire dalle coordinate x=8 y x=32 estendendola per 4 byte su ciascuna linea di pixels.

LINEE da 1070 a 1380:

SHSX e SHBSX come per le precedenti routine solo che lo spostamento è fatto verso sinistra.

LINEE da 1420 a 1580:

SHV1 esegue la rotazione verticale verso l'alto delle 1024 locazioni di RAM mentre

LINEE da 1620 a 1880:

si esegue la rotazione verso l'alto della porzione di schermo già citata.

LINEE da 1920 a 2410:

come le precedenti routines ma con la rotazione fatta verso il basso.

LINEE da 2450 a 2480:

sono poche istruzioni che servono per depositare all'indirizzo MEM1 otto bytes che sono la matrice di definizione del carattere grafico presente sui tasti numerici. Con i registri dello Z-80, A e B contenenti il codice di tale carattere si chiama la subroutine CONV (OB3B HEX) che penserà a creare la rispettiva matrice di definizione e porla all'indirizzo contenuto in HL.

LINEE da 2520 a 2590:

è una routine che il BASIC chiama almeno otto volte quanti sono i BIT in un byte. Viene eseguita una rotazione a sinistra (o a destra in base al valore messo all'indirizzo 7A8E HEX) del byte contenuto in MEM. Il risultato 1 o 0, cioè lo stato del flag di carry, viene dato al registro C con B sempe a zero e quindi ritornato al BASIC.

```
1000 REM
1005 REM  HEX-CODE Caricatore
1010 REM
1015 CLEAR 30999
1020 PRINT  FLASH 1;AT 12,12;"ATTENDERE!"
1025 FOR a=31000 TO 31382
1030 READ a$: BEEP .01,50
1035 PRINT  INVERSE 1;AT 14,16;a$
1040 LET h=CODE a$(1)-48
```

```
1045 LET h=h-7*(a$(1)>"@")
1050 LET l=CODE a$(2)-48
1055 LET l=l-7*(a$(2)>"@")
1060 POKE a,16*h+l: NEXT a
1065 PRINT AT 14,11;"Salvataggio!"
1070 SAVE "TRASF"CODE 31000,382
1075 REM
1080 REM  DATI PROGRAMMA L.M.
1085 REM
1090 REM    SUB.VIDEO TRASF
1095 REM
1100 REM       PRIMA PARTE
1105 REM
1110 DATA "11","58","7B","CD","60","79"
1115 DATA "CD","60","79","21","58","7B"
1120 DATA "18","2B","CD","5C","79","2A"
1125 DATA "95","7A","E5","01","58","7D"
1130 DATA "A7","ED","42","E1","D0","01"
1135 DATA "20","00","09","18","16","CD"
1140 DATA "5C","79","ED","4B","95","7A"
1145 DATA "C5","21","58","7B","A7","ED"
1150 DATA "42","E1","D0","A7","01","20"
1155 DATA "00","ED","42","22","95","7A"
1160 DATA "2A","95","7A","11","00","58"
1165 DATA "18","07","ED","5B","95","7A"
1170 DATA "21","00","58","01","00","02"
1175 DATA "ED","B0","C9","CD","5C","79"
1180 DATA "DD","21","57","7F","0E","20"
1185 DATA "DD","66","00","06","20","DD"
1190 DATA "7E","FF","DD","77","00","DD"
1195 DATA "2B","10","F6","DD","74","01"
1200 DATA "0D","20","EB","CD","54","79"
1205 DATA "01","08","20","C5","05","CD"
1210 DATA "AA","22","E5","06","04","A7"
1215 DATA "CB","1E","23","10","FB","E1"
1220 REM
1225 REM        SECONDA PARTE
1230 REM
1235 DATA "30","02","CB","FE","C1","10"
1240 DATA "EA","C9","CD","5C","79","DD"
1245 DATA "21","58","7B","0E","20","DD"
1250 DATA "66","00","06","20","DD","7E"
1255 DATA "01","DD","77","00","DD","23"
```

```
1260 DATA "10","F6","DD","74","FF","0D"
1265 DATA "20","EB","CD","54","79","01"
1270 DATA "20","20","C5","05","CD","AA"
1275 DATA "22","E5","06","04","A7","CB"
1280 DATA "16","2B","10","FB","E1","30"
1285 DATA "02","CB","C6","C1","10","EA"
1290 DATA "C9","CD","5C","79","11","20"
1295 DATA "00","DD","21","58","7B","4B"
1300 DATA "DD","E5","43","DD","66","00"
1305 DATA "DD","7E","20","DD","77","00"
1310 DATA "DD","19","10","F6","DD","74"
1315 DATA "E0","DD","E1","DD","23","0D"
1320 DATA "20","E6","CD","54","79","01"
1325 DATA "08","1F","3E","04","08","C5"
1330 DATA "CD","AA","22","C1","C5","7E"
1335 DATA "F5","EB","C5","05","CD","AA"
1340 DATA "22","C1","7E","12","10","F5"
1345 REM
1350 REM        TERZA PARTE
1355 REM
1360 DATA "F1","77","C1","79","C6","08"
1365 DATA "4F","08","3D","20","E1","C9"
1370 DATA "CD","5C","79","11","E0","FF"
1375 DATA "DD","21","38","7F","0E","20"
1380 DATA "DD","E5","06","20","DD","66"
1385 DATA "00","DD","7E","E0","DD","77"
1390 DATA "00","DD","19","10","F6","DD"
1395 DATA "74","20","DD","E1","DD","23"
1400 DATA "0D","20","E5","CD","54","79"
1405 DATA "01","08","00","3E","04","08"
1410 DATA "C5","CD","AA","22","C1","C5"
1415 DATA "7E","F5","EB","C5","04","CD"
1420 DATA "AA","22","C1","7E","12","04"
1425 DATA "78","FE","1F","20","F1","F1"
1430 DATA "77","C1","79","C6","08","4F"
1435 DATA "08","3D","20","DD","C9","06"
1440 DATA "8F","78","21","01","5B","C3"
1445 DATA "3B","0B","AF","4F","47","3A"
1450 DATA "00","5B","17","32","00","5B"
1455 DATA "CB","11","C9","58","7B"


7918          0000        ORG  31000
```

```
                 0010 ;
                 0020 ;                "Routine"
                 0030 ;                "Sistema"
                 0040 ;
22AA             0050 BITC   EQU   22AAH
0B3B             0060 CONV   EQU   0B3BH
                 0070 ;
                 0080 ;                "Variab."
                 0090 ;
7B58             0100 INIZ   EQU   31576
0200             0110 COST   EQU   0200H
7D58             0120 FINE   EQU   INIZ+COST
5800             0130 VIDE   EQU   5800H
0020             0140 BITE   EQU   0020H
5B00             0150 MEM    EQU   5B00H
5B01             0160 MEM1   EQU   MEM+1
                 0170 ;
                 0180 ;
                 0190 ;
7918 11587B      0200 SET    LD    DE,INIZ
791B CD6079      0210        CALL  MEMO
791E CD6079      0220        CALL  MEMO
7921 21587B      0230 HOME   LD    HL,INIZ
7924 182B        0240        JR    TRAS
                 0250 ;
                 0260 ;
                 0270 ;
7926 CD5C79      0280 PLUS   CALL  VRAM
7929 2A957A      0290        LD    HL,(RAM)
792C E5          0300        PUSH  HL
792D 01587D      0310        LD    BC,FINE
7930 A7          0320        AND   A
7931 ED42        0330        SBC   HL,BC
7933 E1          0340        POP   HL
7934 D0          0350        RET   NC
7935 012000      0360        LD    BC,BITE
7938 09          0370        ADD   HL,BC
7939 1816        0380        JR    TRAS
                 0390 ;
                 0400 ;
                 0410 ;
793B CD5C79      0420 MINUS  CALL  VRAM
793E ED4B957A    0430        LD    BC,(RAM)
```

```
7942 C5         0440        PUSH BC
7943 21587B     0450        LD   HL,INIZ
7946 A7         0460        AND  A
7947 ED42       0470        SBC  HL,BC
7949 E1         0480        POP  HL
794A D0         0490        RET  NC
794B A7         0500        AND  A
794C 012000     0510        LD   BC,BITE
794F ED42       0520        SBC  HL,BC
                0530 ;
                0540 ;
                0550 ;
7951 22957A     0560 TRAS   LD   (RAM),HL
                0570 ;
7954 2A957A     0580 RAMV   LD   HL,(RAM)
7957 110058     0590        LD   DE,VIDE
795A 1807       0600        JR   LDBC
                0610 ;
                0620 ;
                0630 ;
795C ED5B957A   0640 VRAM   LD   DE,(RAM)
7960 210058     0650 MEMO   LD   HL,VIDE
7963 010002     0660 LDBC   LD   BC,COST
7966 EDB0       0670        LDIR
7968 C9         0680        RET
                0690 ;
                0700 ;
                0710 ;
7969 CD5C79     0720 SHDX   CALL VRAM
796C DD21577F   0730        LD   IX,INIZ+03
FFH
7970 0E20       0740        LD   C,20H
7972 DD6600     0750 LB0    LD   H,(IX+0)
7975 0620       0760        LD   B,20H
7977 DD7EFF     0770 LB1    LD   A,(IX+0FFH
)
797A DD7700     0780        LD   (IX+0),A
797D DD2B       0790        DEC  IX
797F 10F6       0800        DJNZ LB1
7981 DD7401     0810        LD   (IX+1),H
7984 0D         0820        DEC  C
7985 20EB       0830        JR   NZ,LB0
7987 CD5479     0840        CALL RAMV
```

```
                 0850 ;
                 0860 ;
                 0870 ;
798A 010820      0880 SHBDX LD   BC,2008H
798D C5          0890 LB2   PUSH BC
798E 05          0900       DEC  B
798F CDAA22      0910       CALL BITC
7992 E5          0920       PUSH HL
7993 0604        0930       LD   B,4
7995 A7          0940       AND  A
7996 CB1E        0950 LB3   RR   (HL)
7998 23          0960       INC  HL
7999 10FB        0970       DJNZ LB3
799B E1          0980       POP  HL
799C 3002        0990       JR   NC,LB4
799E CBFE        1000       SET  7,(HL)
79A0 C1          1010 LB4   POP  BC
79A1 10EA        1020       DJNZ LB2
79A3 C9          1030       RET
                 1040 ;
                 1050 ;
                 1060 ;
79A4 CD5C79      1070 SHSX  CALL VRAM
79A7 DD21587B    1080       LD   IX,INIZ
79AB 0E20        1090       LD   C,20H
79AD DD6600      1100 LB5   LD   H,(IX+0)
79B0 0620        1110       LD   B,20H
79B2 DD7E01      1120 LB6   LD   A,(IX+1)
79B5 DD7700      1130       LD   (IX+0),A
79B8 DD23        1140       INC  IX
79BA 10F6        1150       DJNZ LB6
79BC DD74FF      1160       LD   (IX+0FFH),
H
79BF 0D          1170       DEC  C
79C0 20EB        1180       JR   NZ,LB5
79C2 CD5479      1190       CALL RAMV
                 1200 ;
                 1210 ;
                 1220 ;
79C5 012020      1230 SHBSX LD   BC,2020H
79C8 C5          1240 LB7   PUSH BC
79C9 05          1250       DEC  B
79CA CDAA22      1260       CALL BITC
```

```
79CD E5         1270        PUSH HL
79CE 0604       1280        LD   B,4
79D0 A7         1290        AND  A
79D1 CB16       1300 LB8    RL   (HL)
79D3 2B         1310        DEC  HL
79D4 10FB       1320        DJNZ LB8
79D6 E1         1330        POP  HL
79D7 3002       1340        JR   NC,LB9
79D9 CBC6       1350        SET  0,(HL)
79DB C1         1360 LB9    POP  BC
79DC 10EA       1370        DJNZ LB7
79DE C9         1380        RET
                1390 ;
                1400 ;
                1410 ;
79DF CD5C79     1420 SHV1   CALL VRAM
79E2 112000     1430        LD   DE,20H
79E5 DD21587B   1440        LD   IX,INIZ
79E9 4B         1450        LD   C,E
79EA DDE5       1460 LB10   PUSH IX
79EC 43         1470        LD   B,E
79ED DD6600     1480        LD   H,(IX+0)
79F0 DD7E20     1490 LB11   LD   A,(IX+20H)
79F3 DD7700     1500        LD   (IX+0),A
79F6 DD19       1510        ADD  IX,DE
79F8 10F6       1520        DJNZ LB11
79FA DD74E0     1530        LD   (IX+0E0H),
H
79FD DDE1       1540        POP  IX
79FF DD23       1550        INC  IX
7A01 0D         1560        DEC  C
7A02 20E6       1570        JR   NZ,LB10
7A04 CD5479     1580        CALL RAMV
                1590 ;
                1600 ;
                1610 ;
7A07 01081F     1620 SHBV1  LD   BC,1F08H
7A0A 3E04       1630        LD   A,4
7A0C 08         1640 LB12   EX   AF,A'F'
7A0D C5         1650        PUSH BC
7A0E CDAA22     1660        CALL BITC
7A11 C1         1670        POP  BC
7A12 C5         1680        PUSH BC
```

```
7A13 7E       1690          LD   A,(HL)
7A14 F5       1700          PUSH AF
7A15 EB       1710 EXX      EX   DE,HL
7A16 C5       1720          PUSH BC
7A17 05       1730          DEC  B
7A18 CDAA22   1740          CALL BITC
7A1B C1       1750          POP  BC
7A1C 7E       1760          LD   A,(HL)
7A1D 12       1770          LD   (DE),A
7A1E 10F5     1780          DJNZ EXX
7A20 F1       1790          POP  AF
7A21 77       1800          LD   (HL),A
7A22 C1       1810          POP  BC
7A23 79       1820          LD   A,C
7A24 C608     1830          ADD  8
7A26 4F       1840          LD   C,A
7A27 08       1850          EX   AF,A'F'
7A28 3D       1860          DEC  A
7A29 20E1     1870          JR   NZ,LB12
7A2B C9       1880          RET
              1890 ;
              1900 ;
              1910 ;
7A2C CD5C79   1920 SHV2     CALL VRAM
7A2F 11E0FF   1930          LD   DE,0FFE0H
7A32 DD21387F 1940          LD   IX,FINE+01
E0H
7A36 0E20     1950          LD   C,20H
7A38 DDE5     1960 LB13     PUSH IX
7A3A 0620     1970          LD   B,20H
7A3C DD6600   1980          LD   H,(IX+0)
7A3F DD7EE0   1990 LB14     LD   A,(IX+0E0H
)
7A42 DD7700   2000          LD   (IX+0),A
7A45 DD19     2010          ADD  IX,DE
7A47 10F6     2020          DJNZ LB14
7A49 DD7420   2030          LD   (IX+20H),H
7A4C DDE1     2040          POP  IX
7A4E DD23     2050          INC  IX
7A50 0D       2060          DEC  C
7A51 20E5     2070          JR   NZ,LB13
7A53 CD5479   2080          CALL RAMV
              2090 ;
```

```
                  2100 ;
                  2110 ;
7A56 010800       2120 SHB2  LD   BC,0008H
7A59 3E04         2130       LD   A,4
7A5B 08           2140 LB15  EX   AF,A'F'
7A5C C5           2150       PUSH BC
7A5D CDAA22       2160       CALL BITC
7A60 C1           2170       POP  BC
7A61 C5           2180       PUSH BC
7A62 7E           2190       LD   A,(HL)
7A63 F5           2200       PUSH AF
7A64 EB           2210 EX1   EX   DE,HL
7A65 C5           2220       PUSH BC
7A66 04           2230       INC  B
7A67 CDAA22       2240       CALL BITC
7A6A C1           2250       POP  BC
7A6B 7E           2260       LD   A,(HL)
7A6C 12           2270       LD   (DE),A
7A6D 04           2280       INC  B
7A6E 78           2290       LD   A,B
7A6F FE1F         2300       CP   1FH
7A71 20F1         2310       JR   NZ,EX1
7A73 F1           2320       POP  AF
7A74 77           2330       LD   (HL),A
7A75 C1           2340       POP  BC
7A76 79           2350       LD   A,C
7A77 C608         2360       ADD  8
7A79 4F           2370       LD   C,A
7A7A 08           2380       EX   AF,A'F'
7A7B 3D           2390       DEC  A
7A7C 20DD         2400       JR   NZ,LB15
7A7E C9           2410       RET
                  2420 ;
                  2430 ;
                  2440 ;
7A7F 0686         2450 GRAF  LD   B,86H
7A81 78           2460       LD   A,B
7A82 21015B       2470       LD   HL,MEM1
7A85 C33B0B       2480       JP   CONV
                  2490 ;
                  2500 ;
                  2510 ;
7A88 AF           2520 RLA   XOR  A
```

```
7A89 4F          2530          LD    C,A
7A8A 47          2540          LD    B,A
7A8B 3A005B      2550          LD    A,(MEM)
7A8E 17          2560          RLA
7A8F 32005B      2570          LD    (MEM),A
7A92 CB11        2580          RL    C
7A94 C9          2590          RET
                 2600 ;
7A95 587B        2610 RAM      DEFW INIZ
                 2620          END

RAM     7A95
RLA     7A88
GRAF    7A7F
EX1     7A64
LB15    7A5B
SHB2    7A56
LB14    7A3F
LB13    7A38
SHV2    7A2C
EXX     7A15
LB12    7A0C
SHBV1   7A07
LB11    79F0
LB10    79EA
SHV1    79DF
LB9     79DB
LB8     79D1
LB7     79C8
SHBSX   79C5
LB6     79B2
LB5     79AD
SHSX    79A4
LB4     79A0
LB3     7996
LB2     798D
SHBDX   798A
LB1     7977
LB0     7972
SHDX    7969
LDBC    7963
MEMO    7960
VRAM    795C
RAMY    7954
```

```
TRAS    7951
MINUS   793B
PLUS    7926
HOME    7921
SET     7918
MEM1    5B01
MEM     5B00
BITE    0020
VIDE    5800
FINE    7D58
COST    0200
INIZ    7B58
CONV    0B3B
BITC    22AA
#       620B
```

COUNTER

Questo è un programma che ho chiamato così poichè simula in modo molto realistico il «moto» di un contatore, ad esempio un contachilometri.

La tecnica utilizzata consiste nel modificare ripetutamente il puntatore che definisce la disposizione in memoria degli U.D.G. cioè dei caratteri grafici definiti dall'utente.

Occorre dapprima definire come U.D.G. i seguenti caratteri grafici:

```
         U.D.G.
A B C D E F G H I J K
0 1 2 3 4 5 6 7 8 9 0
```

A questo pensa automaticamente il programma andando a prelevare i dati che definiscono tali numeri dalla ROM all'indirizzo 15744 e memorizzandoli a partire dal primo carattere grafico.

Come accennato, per realizzare questa simulazione, si modifica il contenuto del byte basso della variabile di sistema UDG.

Un esempio forse chiarirà un poco le idee:

supponiamo di voler stampare il carattere grafico associato alla lettera "A" (così come viene fatto nelle linee 1125, 1145, 1155 del programma). Questa è un'operazione molto semplice, ma modificando il valore della variabile suddetta, aggiungendole ad esempio 1, le cose cambiano. Sempre stampando l'U.D.G. associato ad "A", il risultato è quello di stampare un carattere la cui matrice di definizione è un una combinazione della matrice originale con parte di quella appartenente al carattere successivo.

Il risultato che si ottiene in questa applicazione è quello di vedere i digits del numero stampato al centro dello schermo ruotare a scatti di un pixel per volta simulando molto bene, come già detto, il "moto" di un contachilometri.

```
1000 REM
1005 REM SIMULAZIONE DEL MOTO
1010 REM   DI UN CONTATORE
1015 REM
1020 LET u=23675: LET p=15744
1025 LET lo=88: LET hi=255
1030 LET mx=lo+72: LET ov=mx+7
1035 LET a=lo: LET b=a: LET c=a
1040 LET d=a: LET r=11: LET t=16
1045 REM
1050 REM     U.D.G. = 65368
1055 REM
1060 POKE u,lo: POKE u+1,hi
1065 REM
1070 LET j=0: FOR i=p TO p+80
1075 POKE USR "a"+j,PEEK i
1080 LET j=j+1: NEXT i
1085 LET j=0: FOR i=p TO p+7
1090 POKE USR "k"+j,PEEK i
1095 LET j=j+1: NEXT i
1100 GO SUB 1185
1105 REM
1110 REM         COUNTER
1115 REM
1120 POKE u,a: LET a=a+1
1125 PRINT AT r,t;"A"
1130 POKE u,b: LET b=b+(a>mx)
1135 PRINT AT r,t-1;"A"
```

```
1140 POKE u,c: LET c=c+(b>mx)
1145 PRINT AT r,t-2;"A"
1150 POKE u,d: LET d=d+(c>mx)
1155 PRINT AT r,t-3;"A"
1160 IF a>ov THEN  LET a=lo
1165 IF b>ov THEN  LET b=lo
1170 IF c>ov THEN  LET c=lo
1175 IF d>ov THEN  LET d=lo
1180 GO TO 1120
1185 PLOT (t-4)*8,177-r*8
1190 DRAW 48,0: DRAW 0,-11
1195 DRAW -48,0: DRAW 0,11
1200 PRINT AT r-2,t-4;"ROLLING"
1205 PRINT AT r+2,t-4;"DIGITS"
1210 RETURN
```

GRAF

Questo programma sarà utile a chi studia i grafici delle funzioni matematiche. Infatti è in grado di plottare su video con ottima approssimazione l'andamento di una funzione matematica entro un intervallo di definizione della funzione stessa impostato dall'utente.

Il programma chiede tre parametri:

1) la funzione y = f(x) che si desidera studiare;
2) la definizione di calcolo (numero dei punti in cui si deve calcolare la funzione);
3) l'intervallo di studio.

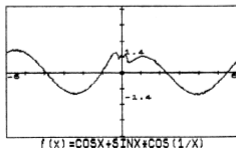
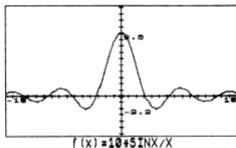
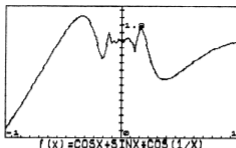
Maggiore è il numero dei punti in cui si calcola la funzione, migliore è la definizione, cioè la precisione con cui verrà disegnata la curva sul video. Un valore accettabile è 40-50 punti, ma questo varia col tipo di funzione e intervallo di studio.

Le caratteristiche di questo programma sono diverse:

1) scalatura automatica;
2) posizionamento assi automatico;
3) il programma deve funzionare, salvo modifiche, in unione ad una parte in L/M in grado di generare caratteri di diverse dimensioni in qualsiasi punto del video. Questa parte in L/M è fornita con ogni Spectrum nella cassetta in dotazione, e, in genere, si trova in coda ad ogni programma di tale cassetta.

Dopo aver digitato GRAF e salvato su nastro in ogni caso, sarà necessario eseguire un CLEAR 32255 e caricare il codice macchina del programma citato.

Per finire salvare il tutto con un GOTO 1775.

f(X)=COSX+SINX*COS(1/X)

```
1000 CLS : LET n=0
1005 INPUT "F(x) = "; LINE y$
1010 INPUT "Definizione del calcolo = "; LI
NE p$
1015 IF p$<>"" THEN  LET n=VAL p$
1020 IF n<10 THEN  GO TO 1010
1025 DIM a(n): DIM b(n)
1030 DIM x(n): DIM y(n)
1035 DEF FN y(x)=VAL y$
1040 INPUT "xmin =";x1;"  xmax = ";x2
1045 IF x2<=x1 THEN  GO TO 1040
1050 LET dx=x2-x1: LET sx=254/dx
1055 FOR i=1 TO n: LET x(i)=x1+(i-1)*dx/(n-
1)+.001
1060 LET y(i)=FN y(x(i)): NEXT i
1065 BEEP .2,20
1070 LET y1=FN y(x(1)): LET y2=FN y(x(1))
1075 FOR i=1 TO n
1080 IF y(i)<y1 THEN  LET y1=y(i)
1085 IF y(i)>y2 THEN  LET y2=y(i)
1090 NEXT i: BEEP .2,20
1095 LET dy=y2-y1: LET sy=174/dy
1100 LET s=sy: IF dx/dy>254/174 THEN  LET s
=sx
1105 LET dx=dx*s: LET dy=dy*s
1110 LET x3=127-dx/2
1115 LET y3=87-dy/2
1120 FOR i=1 TO n
```

```
1125 LET a(i)=(x(i)-x1)*s+x3
1130 LET b(i)=(y(i)-y1)*s+y3
1135 NEXT i
1140 LET x4=a(1)-x1*s
1145 LET y4=y3-y1*s
1150 GO SUB 1360
1155 GO SUB 1455: GO SUB 1500
1160 PLOT 0,0: DRAW 0,175: DRAW 255,0
1165 DRAW 0,-175: DRAW -255,0
1170 FOR i=1 TO n-1
1175 PLOT a(i),b(i)
1180 DRAW a(i+1)-a(i),b(i+1)-b(i)
1185 NEXT i: FLASH 1
1190 GO SUB 1210: FLASH 0: PAUSE 0
1195 INPUT "Cls ? "; LINE c$
1200 IF c$="" OR c$="s" OR c$="S" THEN  CLS
1205 GO TO 1010
1210 REM
1215 REM Sub CHR$ Generatore
1220 REM
1225 LET p$="f(x)=": GO SUB 1270
1230 LET yy=176: LET xs=1: LET ys=2
1235 LET xx=(256-8*xs*LEN p$)/2
1240 GO SUB 1680
1245 LET i=23306: POKE i,xx: POKE i+1,yy
1250 POKE i+2,xs: POKE i+3,ys: POKE i+4,8
1255 LET i=i+4: FOR u=1 TO LEN p$
1260 POKE i+u,CODE p$(u): NEXT u
1265 POKE i+u,255: LET w=USR 32256: RETURN
1270 REM
1275 REM Sub Trova Funzioni
1280 REM
1285 FOR u=1 TO LEN y$
1290 LET b$=y$(u): LET w=CODE b$
1295 IF w=165 THEN  LET b$="RND"
1300 IF w=167 THEN  LET b$="PI"
1305 IF w<178 THEN  GO TO 1325
1310 RESTORE 1350
1315 FOR i=0 TO 11-(189-w)
1320 READ b$: NEXT i
1325 LET p$=p$+b$
1330 NEXT u: RETURN
1335 REM
```

```
1340 REM  DATA Funzioni
1345 REM
1350 DATA "SIN","COS","TAN","ASN","ACS","AT
N"
1355 DATA "LN","EXP","INT","SQR","SGN","ABS
"
1360 REM
1365 REM Sub Stampa Ymax & Ymin
1370 REM
1375 LET xs=1: LET ys=1
1380 LET y1=INT (y1*10)/10
1385 LET y2=INT (y2*10)/10
1390 LET xx=x4+2
1395 LET yy=175-(y4+s*y1)
1400 LET p$=STR$ y1: GO SUB 1415
1405 LET yy=175-(y4+s*y2)
1410 LET p$=STR$ y2
1415 IF yy>165 THEN  LET yy=165
1420 IF yy<0 THEN  LET yy=2
1425 IF xx<0 THEN  LET xx=0
1430 IF xx+8*LEN p$>255 THEN  LET xx=255-8*
LEN p$
1435 GO TO 1245
1440 REM
1445 REM Sub Stampa Xmax & Xmin
1450 REM
1455 LET x1=INT (x1*10)/10
1460 LET x2=INT (x2*10)/10
1465 LET xx=a(1): LET yy=177-y4
1470 LET p$=STR$ x1: GO SUB 1415
1475 LET xx=a(n)
1480 LET p$=STR$ x2: GO TO 1415
1485 REM
1490 REM Sub    Parametri
1495 REM
1500 LET x5=INT x4
1505 LET y5=INT y4
1510 LET xx=x5-3: LET yy=171-y5
1515 IF xx<2 OR xx>255 THEN  GO TO 1530
1520 IF yy<2 OR yy>167 THEN  GO TO 1530
1525 LET p$="o": GO SUB 1415
1530 LET x5=ABS x5: LET y5=ABS y5
1535 IF y5>175 THEN  LET y5=y5-INT (y5/176)
```

```
*176
1540 IF x5>255 THEN  LET x5=x5-INT (x5/255)
*255
1545 LET r=s: IF r>100 OR r<5 THEN  LET r=1
0
1550 REM
1555 REM Sub Stampa Asse Y
1560 REM
1565 IF x4<0 OR x4>255 THEN  GO TO 1630
1570 PLOT x5,0: DRAW 0,175
1575 FOR u=y5 TO 175 STEP r
1580 PLOT x5,u: DRAW 2*(x5<254),0
1585 PLOT x5,u: DRAW -2*(x5>1),0
1590 NEXT u
1595 FOR u=y5 TO 0 STEP -r
1600 PLOT x5,u: DRAW 2*(x5<254),0
1605 PLOT x5,u: DRAW -2*(x5>1),0
1610 NEXT u
1615 REM
1620 REM Sub Stampa Asse X
1625 REM
1630 IF y4<0 OR y4>175 THEN  RETURN
1635 PLOT 0,y5: DRAW 255,0
1640 FOR u=x5 TO 255 STEP r
1645 PLOT u,y5: DRAW 0,2*(y5<174)
1650 PLOT u,y5: DRAW 0,-2*(y5>1)
1655 NEXT u
1660 FOR u=x5 TO 0 STEP -r
1665 PLOT u,y5: DRAW 0,2*(y5<174)
1670 PLOT u,y5: DRAW 0,-2*(y5>1)
1675 NEXT u: RETURN
1680 REM
1685 REM Sub Trova LEN p$
1690 REM
1695 IF LEN p$<33 THEN  RETURN
1700 LET y$=1: LET b$=p$(33 TO )
1705 LET xx=0: LET p$=p$( TO 32)
1710 GO SUB 1245: LET p$=b$
1715 LET yy=184: RETURN
1720 REM
1725 REM      AUTOSTART
1730 REM
1735 INK 1: PAPER 1: BORDER 1
```

```
1740 OVER 0: INVERSE 0
1745 CLEAR 32255
1750 LOAD "CHR$ "CODE 32256
1755 INK 7: CLS : RUN
1760 REM
1765 REM      SALVATAGGIO
1770 REM
1775 SAVE "GRAF f(x)" LINE 1735
1780 SAVE "CHR$ "CODE 32256,300
1785 STOP
```

PARTE III

I GIOCHI

I giochi al computer sono cosa frivola se paragonati ai ben più seri compiti cui viene destinato un elaboratore.

Tuttavia è mia opinione che essi rappresentino un valido aiuto per coloro che desiderino imparare a programmare, e per un valido motivo.

Quando ci si avvicina ad un elaboratore, specialmente se si è alle prime armi, si cerca di instaurare un rapporto più che simpatico con ciò che sino a poco tempo fa pareva destinato a pochi eletti, il computer appunto.

Quando si vuole programmare bisogna avere chiaramente in testa ciò che si vuole il computer faccia, così nel programmare un gioco, in generale, l'utente sa già cosa desidera e può fisicamente vedere sul video ciò che via via sta realizzando.

La cosa non è così immediata se si desidera programmare, ad esempio, la gestione di un archivio di dati anche se i problemi di programmazione potrebbero essere gli stessi.

In linea di principio è più semplice seguire lo spostamento di un omino sul video, perchè fisicamente visibile, anzichè il flusso di dati che si trasferiscono su un supporto magnetico.

Seguono ora una serie di giochi che potranno piacervi o anche non piacervi ma senz'altro servono. In essi troverete differenti metodi per risolvere determinati problemi e stimoli per migliorare il gioco stesso secondo le vostre esigenze.

Un avvertimento:

– i caratteri ASCII posti tra apici " " che nel loro insieme non costituiscono

una frase con senso finito sono da considerarsi caratteri grafici e pertanto vanno introdotti nello Spectrum in tale modo.

Un consiglio:

- non scoraggiatevi alle prime difficoltà.

LUNAR LANDER

Lunar lander è un nome molto noto come gioco e si può dire che ce ne sia almeno una versione per ogni tipo di home computer esistente. Questo realizzato per lo Spectrum è molto bello:

- non è spettacolare o particolarmente variopinto, tuttavia è ben realizzato e divertente poichè è il risultato di una felice integrazione tra L/M e Basic.

Questo gioco è costituito da due parti:

1) la parte in Basic che gestisce il gioco in base ai risultati che provengono alla routine in L/M;
2) la parte in L/M è una sorta di routine di SPRITE capace cioè di spostare oggetti sullo schermo e di rilevarne la collisione con qualcosa di estraneo. È qui utilizzata per spostare il nostro Lander per il video secondo coordinate che il programma Basic fornisce.

Per rendere la cosa più complicata, esiste una fascia di "meteoriti" che la nostra navicella deve evitare per allunare sana e salva. Anche questi meteoriti sono spostati pixel per pixel da una parte di questa routine in L/M.

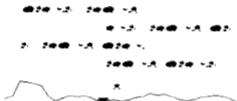
È inclusa la possibilità di generare un suono abbastanza disgustoso qualora l'allunaggio avesse cattivo esito. Suono che molto spesso udirete nel corso del gioco.

Ma veniamo ai comandi:

Tasto	Funzione
1	spinta verso il basso

2	posizione di equilibrio
3	spinta verso l'alto
8	spinta a sinistra
9	posizione normale
0	spinta a destra

Come sempre avendo a che fare con codici in linguaggio macchina occorre prestare molta attenzione nel digitarli e comunque salvare su nastro il programma prima del RUN.

Infine sconsiglio di cambiare i numeri di linea poichè, come ben si nota dal listato alla linea 1070, il L/M indirizza il Basic in punti ben precisi del programma.

```
1000 CLEAR 32299: GO SUB 1350
1005 LET diff=1: LET a=32300: LET b=a+1
1010 LET c=a+2: LET d=a+8: LET puff=a+93
1015 GO SUB 1180
1020 LET a$=INKEY$
1025 IF a$="0" THEN  GO SUB 1075
1030 IF a$="1" THEN  GO SUB 1080
1035 IF a$="2" THEN  GO SUB 1085
1040 IF a$="3" THEN  GO SUB 1090
1045 IF a$="8" THEN  GO SUB 1095
1050 IF a$="9" THEN  GO SUB 1100
1055 LET sx=sx+ax: LET sy=sy+ay
1060 LET x=x+sx: LET y=y+sy
1065 POKE a,x: POKE b,192-y
1070 GO TO USR d
1075 LET ax=q: POKE c,1+8: RETURN
1080 LET ay=-q: RETURN
1085 LET ay=0: RETURN
```

```
1090 LET ay=q: RETURN
1095 LET ax=-q: POKE c,l+16: RETURN
1100 LET ax=0: POKE c,l: RETURN
1105 IF y<8 THEN  LET y=8
1110 IF y>191 THEN  LET y=191
1115 LET sy=-sy: GO TO 1065
1120 IF x<0 THEN  LET x=0
1125 IF x>247 THEN  LET x=247
1130 LET sx=-sx: GO TO 1065
1135 PRINT AT 0,0;
1140 IF (sy*sy+sx*sx)>diff OR y>40 THEN
 POKE c,l+24: RANDOMIZE USR puff: RANDOM
IZE USR d: RANDOMIZE USR puff: PRINT "Di
strutto": LET punti=200-INT y: GO TO 116
0
1145 IF (x>106 OR x<103) THEN  PRINT "Ma
ncato": FOR i=1 TO 4: FOR j=20 TO 12 STE
P -2: BEEP .01,j: NEXT j: NEXT i: PLOT 1
08,4: DRAW 0,24: PRINT AT 17,2;"E'qui ch
e bisogna atterrare";AT 1,0: LET punti=4
00-INT (ABS (x-104.5)): GO TO 1160
1150 BEEP .2,20: BEEP .3,10: LET t0=INT
((FN t(23672)-t0)/50): PRINT "O.k. in ";
t0;" Secondi": LET punti=INT (400+2000/t
0)
1155 POKE c,l: RANDOMIZE USR d
1160 PRINT "Punti= ";punti
1165 PRINT "Premi un tasto"
1170 PAUSE 1: GO SUB 1280: PAUSE 0
1175 CLS : BORDER 1: GO TO 1005
1180 CLS : LET q=.05: RESTORE 1275
1185 PLOT 0,4: FOR x=0 TO 247 STEP 8
1190 READ y: DRAW 8,y: NEXT x
1195 DRAW 7,-4: POKE c+2,0: POKE c+3,0
1200 LET l=PEEK 23675: LET h=PEEK 23676
1205 POKE c,l: POKE c+1,h
1210 POKE c+4,l: POKE c+5,h
1215 LET a$="EF IB EFGH IG  GH"
1220 FOR i=6 TO 15 STEP 3
1225 PRINT AT i,8;a$(i+1 TO );a$( TO i):
 NEXT i
1230 PLOT 101,2: DRAW 2,2
1235 PLOT 103,5: DRAW 10,0
```

```
1240 PLOT 104,4: DRAW 8,0
1245 PLOT 105,3: DRAW 6,0
1250 PLOT 103,2: DRAW 10,0
1255 PLOT 115,2: DRAW -2,2
1260 LET x=32+200*RND: LET y=192-10*RND:
 LET sx=-.5*RND: LET sy=0: LET ax=sy: LE
T ay=sy
1265 DEF FN t(a)=PEEK a+256*PEEK (a+1)+6
5576*PEEK (a+2)
1270 LET t0=FN t(23672): RETURN
1275 DATA 3,20,-1,-2,-3,-14,-6,4,3,-1,1,
-3,-4,0,2,-2,2,3,2,4,-2,1,-3,-4,-2,2,3,2
,1,-1,-4,-3
1280 PRINT ''"1 SPINTA IN BASSO"
1285 PRINT "2 EQUILIBRIO"
1290 PRINT "3 SPINTA IN ALTO"
1295 PRINT "8 SPINTA A SINISTRA"
1300 PRINT "9 POSIZIONE NORMALE"
1305 PRINT "0 SPINTA A DESTRA"
1310 RETURN
1315 POKE 32442,0: POKE 32443,237
1320 POKE 32444,103: POKE 32419,0
1325 POKE 32423,237: POKE 32424,111
1330 RETURN
1335 REM
1340 REM  HEX-CODE Caricatore
1345 REM
1350 RESTORE 1425: GO SUB 1280
1355 PRINT AT 0,10;"ATTENDERE"
1360 FOR i=1 TO 2: READ a,b
1365 FOR j=a TO b: READ a$
1370 LET h=CODE a$(1)-48
1375 LET h=h-7*(a$(1)>"@")
1380 LET l=CODE a$(2)-48
1385 LET l=l-7*(a$(2)>"@")
1390 POKE j,16*h+l: BEEP .01,69
1395 NEXT j: NEXT i: RETURN
1400 REM
1405 REM  DATI PROGRAMMA L.M.
1410 REM
1415 REM LANDER SPRITE ROUTINE
1420 REM
1425 DATA      32306,32599
```

```
1430 DATA "00","00","2A","30","7E","ED"
1435 DATA "5B","32","7E","CD","3D","7F"
1440 DATA "2A","2C","7E","01","51","04"
1445 DATA "7C","D6","B8","D0","01","60"
1450 DATA "04","7D","D6","F8","D0","22"
1455 DATA "30","7E","ED","5B","2E","7E"
1460 DATA "ED","53","32","7E","D5","CD"
1465 DATA "CC","7E","D1","01","FC","03"
1470 DATA "28","03","01","6F","04","C5"
1475 DATA "D5","E5","21","C0","40","CD"
1480 DATA "B0","7E","21","80","48","CD"
1485 DATA "B0","7E","21","20","48","CD"
1490 DATA "9C","7E","21","E0","48","CD"
1495 DATA "9C","7E","E1","D1","CD","26"
1500 DATA "7F","C1","C9","0E","00","21"
1505 DATA "C4","7E","06","00","10","FE"
1510 DATA "7E","D3","FE","23","0D","20"
1515 DATA "F5","C9","00","00","11","20"
1520 DATA "00","0E","08","7E","19","17"
1525 DATA "06","20","2B","CB","16","10"
1530 DATA "FB","24","0D","20","F2","C9"
1535 DATA "7D","F6","1F","5F","0E","08"
1540 DATA "06","20","54","1A","1F","CB"
1545 DATA "1E","23","10","FB","7D","D6"
1550 DATA "20","6F","24","0D","20","EE"
1555 DATA "C9","00","00","00","E5","CD"
1560 DATA "E6","7E","0E","08","C5","CD"
1565 DATA "0A","7F","A6","20","04","79"
1570 DATA "23","A6","2B","C1","20","03"
1575 DATA "0D","20","EF","E1","C9","00"
1580 DATA "D5","55","CB","3D","CB","3D"
1585 DATA "CB","3D","7C","4F","0F","0F"
1590 DATA "0F","E6","18","47","7C","E6"
1595 DATA "07","B0","C6","40","67","79"
1600 DATA "87","87","E6","E0","85","6F"
1605 DATA "7A","E6","07","47","D1","C9"
1610 DATA "24","3E","07","A4","20","0A"
1615 DATA "7D","C6","20","6F","38","04"
1620 DATA "7C","C6","F8","67","AF","4F"
1625 DATA "B0","1A","13","C8","1F","CB"
1630 DATA "19","10","FB","C9","E5","CD"
1635 DATA "E6","7E","0E","08","C5","CD"
1640 DATA "0A","7F","B6","77","23","7E"
```

```
1645 DATA "B1","77","2B","C1","0D","20"
1650 DATA "F1","E1","C9","E5","CD","E6"
1655 DATA "7E","0E","08","C5","CD","0A"
1660 DATA "7F","EE","FF","A6","77","23"
1665 DATA "79","EE","FF","A6","77","2B"
1670 DATA "C1","0D","20","ED","E1","C9"
1675 REM
1680 REM     LANDER U.D.G.
1685 REM
1690 DATA USR "a",USR "i"+7
1695 DATA "00","3C","7E","7E","3C","3C"
1700 DATA "42","C3","1C","3E","7E","7E"
1705 DATA "3E","44","C2","03","38","7C"
1710 DATA "7E","7E","7C","22","43","C0"
1715 DATA "24","81","5A","3C","3C","5A"
1720 DATA "81","24","01","0F","1F","3F"
1725 DATA "3F","3F","1F","07","F0","F8"
1730 DATA "FC","FC","FC","F8","F8","30"
1735 DATA "18","7C","7C","3C","09","61"
1740 DATA "F0","60","00","1C","FF","FF"
1745 DATA "FF","FE","FC","0C","00","00"
1750 DATA "C0","F8","70","00","03","03"
```

BASE ATTACK

È un gioco molto semplice ma divertente. Lo scopo è quello di evitare che la solita astronave aliena atterri distruggendo le basi misilistiche. Solo che è abbastanza difficile riuscirci perchè queste basi hanno pochi missili a loro disposizione e una volta esauriti non c'è più nulla da fare.

È questione di abilità:

– ci sono tre basi, ciascuna con i suoi cinque missili che si possono far partire premendo i seguenti tasti:

  B per la base posta a sinistra

  N per la base posta al centro

M per la base posta a destra

il mirino è invece spostabile sullo schermo agendo sui tasti W, A, S, Z secondo direzioni ovviamente intuibili sulla tastiera.

Anche qui come in molti listati di questo libro si fa largo uso di caratteri grafici. Attenzione quindi alle lettere poste tra apici che, ove non abbiano un senso compiuto, son da associarsi al carattere grafico omologo.

PUNTI=300

```
1000 REM
1005 REM     BASE  ATTACK
1010 REM
1015 BORDER 6: PAPER 5: INK 0: CLS
1020 PRINT AT 6,12; FLASH 1;"MISSILE"
1025 PRINT AT 7,12; INVERSE 1; FLASH 1;"CO
MMAND"
1030 PRINT AT 11,0; INVERSE 1;"<W,A,S,Z> p
er muovere il Mirino "
1035 PRINT AT 13,0; INVERSE 1;"<B,N,M> per
 sparare con le Basi "
1040 GO SUB 1275
1045 LET a$="": FOR i=0 TO 31
1050 LET a$=a$+"■": NEXT i
1055 PAUSE 0: POKE 23658,0
1060 BORDER 0: PAPER 0: INK 7: FLASH 0: BR
IGHT 0: CLS
1065 PRINT AT 18,0; INK 5;"   BC    "; INK
```

```
7;"EEEE"; INK 5;"   BC   "; INK 7;"EEEE";
INK 5;"   BC   "
1070 PRINT AT 19,0; INK 6;" A■"; INK 5;"JK
"; INK 6;"■I"; INK 3;" DDDD"; INK 6;" A■";
 INK 5;"JK"; INK 6;"■I"; INK 3;" DDDD"; IN
K 6;" A■"; INK 5;"JK"; INK 6;"■I "
1075 PRINT AT 20,0; INK 6;"A■■■■■■I"; INK
3;"DDDD"; INK 6;"A■■■■■■I"; INK 3;"DDDD";
INK 6;"A■■■■■■I"
1080 PRINT AT 21,0; INK 2;a$
1085 LET x=10: LET y=16
1090 LET sk=5: LET sco=0
1095 LET nn=11: LET na=5
1100 LET nb=5: LET nc=5
1105 LET a=0: LET b=INT (RND*31): LET nn=n
n-1: IF nn=0 THEN  BEEP 0.3,10: BEEP 0.3,0
: LET nn=10: LET na=5: LET nb=5: LET nc=5:
 LET sk=sk-(sk>0)
1110 PRINT AT a,b; INK 4;"GH"
1115 PRINT AT 0,0; INK 7;"PUNTI=";sco
1120 IF a=18 THEN  GO TO 1250
1125 FOR q=1 TO sk
1130 PRINT  INK 7;AT x,y;"F"
1135 PRINT AT 20,3;na;AT 20,15;nb;AT 20,27
;nc: LET rr=0
1140 IF INKEY$="b" OR INKEY$="n" OR INKEY$
="m" THEN  GO TO 1220
1145 IF INKEY$="a" THEN  PRINT AT x,y;" ":
 LET y=y-(y>0)
1150 IF INKEY$="s" THEN  PRINT AT x,y;" ":
 LET y=y+(y<31)
1155 IF INKEY$="w" THEN  PRINT AT x,y;" ":
 LET x=x-(x>1)
1160 IF INKEY$="z" THEN  PRINT AT x,y;" ":
 LET x=x+(x<17)
1165 NEXT q
1170 LET c=INT (RND*3)+1
1175 LET a=a+(c=1)
1180 LET a=a+(c=2)
1185 LET b=b+(c=2)
1190 LET a=a+(c=3)
1195 LET b=b-(c=3)
1200 IF b<1 THEN  LET b=1
```

```
1205 IF b>30 THEN  LET b=30
1210 PRINT AT a-1,b-1;"    "
1215 GO TO 1110
1220 IF INKEY$="b" AND na>0 THEN  INK 6: L
ET rr=1: PLOT 32,33: DRAW (8*y)-28,(8*(21-
x))-29: BEEP 0.01,10: BEEP 0.01,0: BEEP 0.
1,-10: PLOT  OVER 1;32,33: DRAW  OVER 1;(8
*y)-28,(8*(21-x))-29: INK 7: PRINT AT x,y;
"F": LET na=na-1
1225 IF INKEY$="n" AND nb>0 THEN  INK 6: L
ET rr=1: PLOT 128,33: DRAW (8*y)-124,(8*(2
1-x))-29: BEEP 0.01,10: BEEP 0.01,0: BEEP
0.1,-10: PLOT  OVER 1;128,33: DRAW  OVER 1
;(8*y)-124,(8*(21-x))-29: INK 7: PRINT AT
x,y;"F": LET nb=nb-1
1230 IF INKEY$="m" AND nc>0 THEN  INK 6: L
ET rr=1: PLOT 224,33: DRAW (8*y)-220,(8*(2
1-x))-29: BEEP 0.01,10: BEEP 0.01,0: BEEP
0.1,-10: PLOT  OVER 1;224,33: DRAW  OVER 1
;(8*y)-220,(8*(21-x))-29: INK 7: PRINT AT
x,y;"F": LET nc=nc-1
1235 IF rr<>1 THEN  GO TO 1145
1240 IF ATTR (a,b)=7 OR ATTR (a,b+1)=7 THE
N  PRINT AT a,b;"  ": FOR d=0 TO 5: PRINT
AT a-d*((a-d)<17),b+d*((b+d)<31);"*";AT a-
d*2*((a-d*2)<17),b-d*2*((b-d*2)<31);"+": B
EEP .001,d*5: PRINT AT a-d*((a-d)<17),b+d*
((b+d)<31);" ";AT a-2*d*((a-d*2)<17),b-d*2
*((b-2*d)<31);" ": NEXT d: LET sco=sco+100
: LET rr=0: GO TO 1105
1245 GO TO 1145
1250 PRINT AT a,b; INK 2;"GH"
1255 BEEP 1,0: BEEP 0.6,0: BEEP 0.3,0: BEE
P 1,0: BEEP 1,-5
1260 PRINT AT 12,1; INK 2; PAPER 7; FLASH
1;"INVASORI ATTERRATI -----------"
1265 PRINT "   BATTI UN TASTO PER GIOCARE
": PAUSE 0: PAUSE 0
1270 CLS : GO TO 1060
1275 REM
1280 REM  HEX-CODE Caricatore
1285 REM
1290 FOR a=USR "a" TO USR "k"+7
```

```
1295 READ a$
1300 LET h=CODE a$(1)-48
1305 LET h=h-7*(a$(1)>"@")
1310 LET l=CODE a$(2)-48
1315 LET l=l-7*(a$(2)>"@")
1320 POKE a,16*h+l: NEXT a
1325 RETURN
1330 REM
1335 REM  CHR$ GRAFICI U.D.G.
1340 REM
1345 REM       BASE ATTACK
1350 REM
1355 DATA "01","03","07","0F"
1360 DATA "1F","3F","7F","FF"
1365 DATA "01","01","0F","1D"
1370 DATA "79","71","7F","7F"
1375 DATA "80","80","F0","B8"
1380 DATA "9E","8E","FE","FE"
1385 DATA "7E","66","66","42"
1390 DATA "42","66","66","7E"
1395 DATA "00","00","18","18"
1400 DATA "3C","3C","7E","7E"
1405 DATA "18","18","18","E7"
1410 DATA "E7","18","18","18"
1415 DATA "03","0F","19","39"
1420 DATA "7F","61","60","3C"
1425 DATA "C0","F0","98","9C"
1430 DATA "FE","86","06","3C"
1435 DATA "80","C0","E0","F0"
1440 DATA "F8","FC","FE","FF"
1445 DATA "7F","60","7F","60"
1450 DATA "7F","60","7F","00"
1455 DATA "FE","06","FE","06"
1460 DATA "FE","06","FE","00"
```

PISTA

Ciò che segue è un giochino molto semplice, ma forse anche per questo divertente a giocarsi. È questione di abilità, bisogna guidare un bob seguen-

do un tracciato senza urtare gli ostacoli che questo presenta. Si usa un carattere grafico per disegnare e muovere il bob lungo la pista mentre il tracciato è realizzato con il carattere shift 5. Per muovere il bob utilizzare i tasti W ed E.

```
1000 REM OLIMPIC BOB
1005 REM
1010 GO SUB 1300
1015 REM
1020 REM DEFINIZIONE CARATTERI
1025 REM
1030 REM BOB
1035 REM
1040 POKE USR "a"+0,BIN 00100000
1045 POKE USR "a"+1,BIN 00101000
1050 POKE USR "a"+2,BIN 11101000
1055 POKE USR "a"+3,BIN 11111100
1060 POKE USR "a"+4,BIN 01111110
1065 POKE USR "a"+5,BIN 00111110
1070 POKE USR "a"+6,BIN 00011110
1075 POKE USR "a"+7,BIN 00001110
1080 REM
1085 REM PISTA
1090 REM
1095 REM utilizzato carattere
1100 REM grafico sul tasto 5
1105 REM
1110 LET hy=1
1115 REM
1120 REM TRACCIAMENTO PISTA
1125 REM
1130 PAPER 7: INK 7: CLS
1135 LET x=(RND*10)+5
1140 LET hx=x+2
1145 FOR k=1 TO 20
1150 PRINT AT k,x; INK 0;"▌";AT k,x+z;"▌"
1155 LET x=x+(SGN (RND-.5))
1160 IF x>31 THEN  LET x=31
1165 IF x<0 THEN  LET x=0
1170 NEXT k
1175 PRINT AT hy,hx; INK 2;"A"
1180 PAUSE 50
```

```
1185 FOR w=1 TO 20
1190 LET x=x+(SGN (RND-.5))
1195 IF x>31 THEN  LET x=31
1200 IF x<0 THEN  LET x=0
1205 POKE 23692,-1
1210 PRINT AT 21,x; INK 0;" ▌";AT 21,x+z;"▌
"''
1215 GO SUB 1250
1220 NEXT w
1225 FOR w=1 TO 20
1230 PRINT AT 21,1''
1235 GO SUB 1250
1240 NEXT w
1245 GO SUB 1345
1250 REM
1255 REM MOVIMENTO BOB
1260 REM
1265 LET b$=INKEY$
1270 PRINT AT hy-1,hx;" "
1275 IF b$="q" THEN  LET hx=hx-1
1280 IF b$="w" THEN  LET hx=hx+1
1285 IF ATTR (hy,hx)=56 THEN  GO TO 1355
1290 PRINT AT hy,hx; INK 2;"A"
1295 RETURN
1300 REM
1305 REM ISTRUZIONI
1310 REM
1315 PRINT AT 1,1; INK 1;"USA I TASTI Q &
W PER STERZARE"
1320 PRINT AT 10,10; INK 2; FLASH 1;"OLIMP
IC BOB"
1325 INPUT "LIVELLO DIFFICOLTA' (1-5) ?:";
z
1330 IF z<1 OR z>5 THEN  GO TO 1325
1335 LET z=9-z
1340 RETURN
1345 PRINT AT 20,1; INK 0;"CE L'HAI FATTA
!"
1350 GO TO 1365
1355 PRINT AT 20,1; INK 0;"TI SEI SCHIANTA
TO !"
1360 BEEP 1,-10
1365 INPUT "GIOCHI ANCORA ?:";g$
```

```
1370 IF g$(1)="s" THEN  RUN
1375 INK 0: PAPER 7: BORDER 7: CLS
```

QUASIMODO

Si tratta di un gioco realizzato in Basic ma che presenta qualità non indifferenti. Infatti si sviluppa in 5 fasi di gioco e ognuna ha un suo schermo particolare.

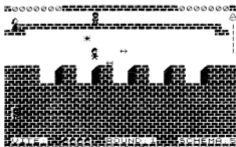
Lo scopo è di raggiungere la fanciulla chiusa nel castello e per far ciò occorre superare innumerevoli insidie. Bisogna fare molto in fretta perchè Quasimodo, il mostro del castello, è geloso e scalerà il muro per raggiungervi e uccidervi.

I comandi sono molto semplici:

1 per muovere a sinistra

2 per muovere a destra

0 per saltare

Il programma per chi possiede uno Spectrum 16k deve essere caricato in due volte:

– prima i dati che definiscono i caratteri U.D.G., cioè dalla linea 2135 (che viene vista come subroutine), poi il programma vero e proprio.

Ciascuno in base alla sua esperienza saprà opportunamente apportare le modifiche necessarie per rendere i due listati un unico blocco.

```
1000 OVER 0: GO SUB 2135: LET hi=0
1005 GO SUB 2060
1010 GO SUB 1990
1015 GO SUB 1620
1020 PRINT AT x,y;"  ";AT x+1,y;"  "
1025 IF INKEY$="" THEN  GO TO 1065
1030 IF INKEY$<>"2" THEN  GO TO 1045
1035 LET m$="A": LET n$="B"
1040 IF y<31 AND ATTR (x+1,y+1)<>86 THEN  L
ET y=y+1
1045 IF INKEY$<>"1" THEN  GO TO 1060
1050 LET m$="C": LET n$="D"
1055 IF y>0 AND ATTR (x+1,y-1)<>86 THEN  LE
T y=y-1
1060 IF INKEY$="0" THEN  GO TO 1100
1065 IF y=31 THEN  GO TO 1525
1070 PRINT AT x,y; INK 5;m$;AT x+1,y;n$
1075 GO SUB 1215
1080 GO SUB 1240+le*30
1085 IF ATTR (x+2,y)<>86 THEN  GO TO 1170
1090 IF ATTR (x,y)<>69 OR ATTR (x+1,y)<>69
THEN  GO TO 1560
1095 GO TO 1020
1100 IF m$="C" THEN  LET n$="F": LET jd=-1:
 GO TO 1115
1105 IF m$="A" THEN  LET n$="E": LET jd=1:
GO TO 1115
1110 LET jd=0
1115 FOR a=-1 TO 1 STEP .5: PRINT AT x,y;"
";
1120 PRINT AT x+1,y;"  ": LET x=x+a
1125 IF y<31 AND y>0 THEN  LET y=y+jd
1130 PRINT AT x,y; INK 5;m$;AT x+1,y;n$
1135 GO SUB 1240+le*30
1140 IF rh=0 AND y=INT (rp/8) OR y+1=INT (r
p/8) THEN  GO TO 1175
```

```
1145 IF y=31 THEN  GO TO 1525
1150 IF le=2 OR le=4 THEN  GO TO 1160
1155 IF ATTR (x,y)<>69 OR ATTR (x+1,y)<>69
THEN  GO TO 1560
1160 BEEP .01,a: NEXT a
1165 IF ATTR (x+2,y)=86 THEN  GO TO 1070
1170 GO TO 1560
1175 LET rh=1: PRINT AT x,y;" ";AT x+1,y;"
"
1180 LET y=INT (rp/8): LET x=8
1185 LET m$="A": LET n$="B"
1190 PRINT AT x,y;" ";AT x+1,y;" ": LET y=r
p/8-1
1195 PRINT AT x,y; INK 5;"A";AT x+1,y;"M"
1200 BEEP .005,y: GO SUB 1215: GO SUB 1300
1205 IF INKEY$="0" THEN  GO TO 1100
1210 GO TO 1190
1215 IF mx<=9 THEN  GO TO 1245
1220 LET mx=mx-(ro/5)
1225 PRINT AT mx,1; PAPER 6; INK 1;"N";AT m
x+1,1;"P"
1230 PRINT AT mx+2,1; INK 6; PAPER 2;"G"
1235 IF mx<=9 THEN  PRINT  INK 6; PAPER 2;A
T 10,1;"G";AT 11,1;"G"
1240 RETURN
1245 PRINT AT 8,1; INK 5;"N";AT 9,1;"O": LE
T mx=mx-1
1250 IF mx>=0 THEN  RETURN
1255 FOR a=2 TO y-1: PRINT AT x+1,a; INK 6;
"H"
1260 BEEP .01,a: BEEP .005,0: BEEP .001,-5
1265 PRINT AT x+1,a;" ": NEXT a: GO TO 1560
1270 PRINT AT 9,ax;" "
1275 PRINT AT 7,29-ax;" "
1280 LET ax=ax-1
1285 IF ax<1 THEN  LET ax=29
1290 PRINT AT 9,ax; INK 6;"Q";AT 7,29-ax;"Q
"
1295 RETURN
1300 PLOT 128,167: DRAW  INVERSE 1;rp-128,-
60
1305 LET rp=rp+rd
1310 IF rp>156 THEN  LET rd=-8
```

```
1315 IF rp<100 THEN  LET rd=8
1320 PLOT 128,167: DRAW rp-128,-60
1325 RETURN
1330 FOR b=6 TO 26 STEP 5
1335 PRINT AT ax,b; INK 4;"R";AT ax-1,b; IN
K 7;" "
1340 NEXT b
1345 LET ax=ax+ad: IF ax=6 THEN  LET ad=1
1350 IF ax=10 THEN  LET ad=-1
1355 RETURN
1360 GO SUB 1300: PRINT AT 10,ax; INK 6; PA
PER 2;"G"
1365 PRINT AT 10,31-ax; INK 6; PAPER 2;"G"
1370 PRINT AT 10,ax+1;" ";AT 10,30-ax;" "
1375 LET ax=ax+ad/2
1380 IF ax=4 OR ax=14 THEN  LET ad=-ad
1385 RETURN
1390 PRINT AT 1,ax; INK 5;" N ";AT 2,ax;" O
 "
1395 LET ax=ax+(ax<27 AND ax<y)-(ax>2 AND a
x>y)
1400 PRINT AT by,bx;" ": LET by=by+1
1405 IF by=9 THEN  LET by=4: LET bx=ax
1410 PRINT AT by,bx; INK 6;"Q"
1415 PRINT AT 9,cy; INK 6;"I ";AT 7,29-cy;"
 H"
1420 LET cy=cy-1
1425 IF cy=0 THEN  PRINT AT 7,29;" ";AT 9,1
;" ": LET cy=29
1430 RETURN
1435 PRINT AT 1,ax+1; INK 5;" "
1440 PRINT AT 2,ax+1; INK 5;" "
1445 FOR a=29 TO 2 STEP -1
1450 PRINT  INK 7;AT 1,a;"C ";AT 2,a;"D "
1455 BEEP .01,a: PAUSE 2
1460 PRINT  INK 7;AT 2,a;"F "
1465 BEEP .01,-a: PAUSE 5: NEXT a
1470 LET sc=sc+(ro*1000)
1475 PRINT AT 7,2;"O.k.! Hai salvato Esmera
lda !!"
1480 PRINT  FLASH 1; INK 0; PAPER 4;AT 15,7
;"Riprovaci ancora !"
1485 FOR a=1 TO 255: OUT 254,-a
```

```
1490 NEXT a: LET ro=ro+1
1495 IF be=4 THEN  LET sc=sc+ro*250: PRINT
AT 17,7;"Super BONUS : ";250*ro
1500 FOR a=1 TO 60 STEP 2
1505 BEEP .001,a: NEXT a
1510 IF ro>4 THEN  LET ro=4
1515 LET le=1: LET be=0
1520 PAUSE 0: PAUSE 0: GO TO 1015
1525 PRINT AT x,y;" ";AT x+1,y;" "
1530 FOR a=7 TO 3 STEP -1
1535 PRINT AT a,31; INK 5;"A";AT a+1,31; IN
K 5;"M"
1540 BEEP .02,30: BEEP .02,25
1545 PRINT AT a,31;" ı";AT a+1,31;" ı": NEXT
a
1550 IF le=5 THEN  GO TO 1435
1555 LET sc=sc+100: LET le=le+1: LET be=be+
1: GO TO 1015
1560 GO SUB 2035: FOR a=255 TO 0 STEP -5
1565 PRINT AT x,y; INK RND*7;m$;AT x+1,y;n$
1570 OUT a,-a: NEXT a: LET be=0
1575 LET li=li-1: IF li=0 THEN  GO TO 1585
1580 GO TO 1015
1585 PRINT AT 11,12; PAPER 2;"FINE GIOCO"
1590 FOR a=50 TO -50 STEP -5
1595 BEEP .01,a: BEEP .01,0
1600 BEEP .005,-5: NEXT a
1605 PRINT AT 13,10; FLASH 1;"PREMI UN TAST
O"
1610 IF sc>hi THEN  LET hi=sc
1615 PAUSE 0: PAUSE 0: GO TO 1005
1620 BORDER 0: PAPER 0: INK 7: BRIGHT 1: CLS
1625 PRINT AT 0,0; INK 6; PAPER 2;a$
1630 PRINT AT 0,1;"0000000";AT 0,24;"000000
0"
1635 PRINT AT 0,8-LEN STR$ sc;sc;AT 0,31-LE
N STR$ hi;hi
1640 PRINT #0;AT 0,0; INK 6; PAPER 2;a$;AT
1,0;a$
1645 FOR a=1 TO li
1650 PRINT #0;AT 0,a+7; INK 4;"A"
1655 NEXT a
1660 PRINT #0;AT 0,1;"VITE:"
```

```
1665 PRINT #0;AT 0,14;"ROUND:";ro
1670 PRINT #0;AT 0,24;"SCHEMA:";le
1675 FOR a=10 TO be
1680 PRINT AT 0,a*2+9; INK 5;"J";AT 1,a*2+9
;"K"
1685 NEXT a
1690 FOR a=10 TO 21
1695 PRINT AT a,0; INK 6; PAPER 2;a$
1700 NEXT a
1705 PRINT AT 1,31;"J";AT 2,31;"K"
1710 FOR a=3 TO 7
1715 PRINT AT a,31; INK 5;" |"
1720 NEXT a
1725 LET rp=300: LET rd=0
1730 GO SUB 1720+le*35
1735 IF le=1 THEN  LET a=31: GO TO 1930
1740 LET rh=0: LET mx=19
1745 LET x=8: LET y=0: LET m$="A": LET n$="
B"
1750 RETURN
1755 FOR a=10 TO 12: FOR b=5 TO 25 STEP 5
1760 PRINT AT a,b;"  "; INK 5;"■"
1765 NEXT b: NEXT a
1770 FOR b=7 TO 27 STEP 5
1775 PRINT AT 10,b; INK 5;"L"
1780 NEXT b: LET ax=29
1785 LET ay=6: RETURN
1790 FOR a=10 TO 19
1795 PRINT AT a,10;b$( TO 11); INK 5;"■"
1800 NEXT a
1805 PRINT AT 20,10; INK 5; PAPER 2; FLASH
1;f$( TO 12)
1810 LET rp=128: LET rd=8
1815 PRINT AT 10,21; INK 5;"L"
1820 RETURN
1825 GO SUB 1755
1830 FOR a=6 TO 26 STEP 5
1835 PRINT AT 11,a;"N"
1840 PRINT AT 12,a;"O"
1845 NEXT a
1850 LET ax=10
1855 LET ad=-1: RETURN
1860 FOR a=10 TO 19
```

```
1865 PRINT AT a,5;b$( TO 21);
1870 PRINT  INK 5;"■": NEXT a
1875 PRINT AT 20,5; INK 5; PAPER 2; FLASH 1
;f$
1880 PRINT  INK 5;AT 10,26;"L"
1885 LET ax=5: LET ad=1: LET rd=8
1890 LET rp=128: RETURN
1895 GO SUB 1755
1900 PRINT AT 3,2; INK 6; PAPER 2;a$( TO 28
)
1905 PRINT AT 2,1;"S";AT 3,1;"T"
1910 PRINT AT 4,0; INK 6; PAPER 2;"GGG";AT
4,29;"GG"
1915 LET cy=29: LET ax=16
1920 LET by=4: LET bx=ax
1925 RETURN
1930 FOR b=-1 TO 1
1935 PRINT AT 19+b,a; INK 5;"C";AT 20+b,a;"
D"
1940 PAUSE 2
1945 PRINT AT 20+b,a; INK 5;"F"
1950 BEEP .05,b
1955 PRINT AT 19+b,a; INK 6; PAPER 2;"G";AT
 20+b,a;"G"
1960 LET a=a-1: IF a THEN  GO TO 1985
1965 FOR b=21 TO 10 STEP -1
1970 PRINT AT b,a; INK 6; PAPER 2;"G";
1975 PRINT AT b-1,a; INK 5;"B";AT b-2,a;"A"
1980 BEEP .05,b: NEXT b: GO TO 1740
1985 NEXT b: GO TO 1930
1990 LET sc=0: LET le=1: LET li=4
1995 LET a$="": LET b$="": LET f$=""
2000 FOR i=1 TO 32
2005 LET a$=a$+"G"
2010 LET b$=b$+" ": NEXT i
2015 FOR i=1 TO 22
2020 LET f$=f$+"U": NEXT i
2025 LET be=0: LET ro=1
2030 RETURN
2035 IF ATTR (x+2,y)<>71 THEN  RETURN
2040 PRINT AT x,y;" ";AT x+1,y;" "
2045 BEEP .02,-y: LET x=x+1: LET y=y+1
2050 PRINT AT x,y;m$;AT x+1,y;n$
```

```
2055 GO TO 2035
2060 BORDER 2: PAPER 6: INK 0
2065 CLS : PRINT AT 19,13;"Ciao !"
2070 PRINT AT 5,10; INVERSE 1;"*QUASIMODO*"
2075 PRINT AT 9,0;"I Comandi sono :"
2080 PRINT AT 11,4;"1 per muovere a sinistr
a"
2085 PRINT AT 12,4;"2 per muovere a destra"
2090 PRINT AT 13,4;"0 per saltare"
2095 PRINT AT 21,1; FLASH 1;"Premi un tasto
 per iniziare !!"
2100 PRINT AT 16,14;"SC"
2105 PRINT AT 17,14;"TE"
2110 PAUSE 10
2115 PRINT AT 17,15;"F"
2120 PAUSE 10
2125 IF INKEY$="" THEN  GO TO 2100
2130 RETURN
2135 REM
2140 REM  HEX-CODE Caricatore
2145 REM
2150 PRINT AT 12,12;"ATTENDI"
2155 FOR a=USR "a" TO USR "u"+7
2160 READ a$
2165 LET h=CODE a$(1)-48
2170 LET h=h-7*(a$(1)>"@")
2175 LET l=CODE a$(2)-48
2180 LET l=l-7*(a$(2)>"@")
2185 POKE a,16*h+l: NEXT a
2190 RETURN
2195 REM
2200 REM  DATI DI DEFINIZIONE
2205 REM
2210 REM   CARATTERI GRAFICI
2215 REM
2220 DATA "38","7C","7E","F4","EC","C2"
2225 DATA "44","78","FA","FF","7E","38"
2230 DATA "38","30","30","38","1C","3E"
2235 DATA "7E","2F","37","43","22","1E"
2240 DATA "5F","FF","7E","1C","0C","0C"
2245 DATA "0C","1C","7C","FC","BA","BF"
2250 DATA "38","7C","44","C6","3E","3F"
2255 DATA "3D","FD","1C","7E","42","C3"
```

```
2260 DATA "FE","FE","FE","00","EF","EF"
2265 DATA "EF","00","00","00","04","C2"
2270 DATA "FF","C2","04","00","00","00"
2275 DATA "21","42","FF","42","21","00"
2280 DATA "00","00","18","24","24","42"
2285 DATA "5A","66","81","91","66","18"
2290 DATA "00","00","00","00","01","03"
2295 DATA "07","0F","1F","3F","7F","FF"
2300 DATA "FA","FF","7E","38","38","1D"
2305 DATA "0F","07","08","7F","3E","7F"
2310 DATA "63","36","7F","08","3E","7B"
2315 DATA "67","7F","36","36","76","07"
2320 DATA "3E","6F","73","7F","36","36"
2325 DATA "37","70","92","54","38","FE"
2330 DATA "38","54","92","00","18","18"
2335 DATA "3C","5A","18","18","3C","18"
2340 DATA "18","3C","74","62","62","E4"
2345 DATA "E8","EE","5F","3E","3C","7E"
2350 DATA "7E","7E","FF","24","08","89"
2355 DATA "5A","55","A5","A9","52","FF"
```

WALL RUN

Ogni secondo che passa, il muro dal basso cresce verso l'alto e gli omini che sono sulla navetta trovano sempre più difficoltà a raggiungere la TERRA.

È questione di tempo e di abilità. Solo con le bombe si può distruggere il muro che avanza e nel momento giusto si possono sganciare gli omini.

Fortunatamente sia le bombe che gli omini si possono guidare nella loro discesa premendo i relativi tasti:

<Z> <X> sgancia e guida sinistra, destra la bomba
<N> <M> sgancia e guida sinistra, destra gli omini.

```
1000 REM
1005 REM        WALL RUN
1010 REM
1015 PAPER 7: BORDER 7: INK 1: CLS
1020 POKE 23658,0: GO SUB 1305
1025 LET hs=0
1030 GO SUB 1205
1035 PRINT AT 0,13;"HI:";hs
1040 LET x=x+f
1045 IF x=0 OR x=29 THEN  BEEP .04,10: LET
f=-f
1050 PRINT AT 0,0; INK 1;"PUNTI:";s;AT 0,24
; INK 1;"VITE:";li;"  "
1055 PRINT AT 1,x; INK 2;" B "
1060 LET i$=INKEY$
1065 IF i$="z" OR i$="x" THEN  IF t<>1 THEN
  LET t=1: LET k=x+1: LET l=2
1070 IF i$="m" OR i$="n" AND x>3 AND x<27 T
HEN  IF z<>1 THEN  LET z=1: LET q=x+1: LET
i=2
1075 IF t<>1 THEN  GO TO 1105
1080 PRINT AT l,k;" ": LET l=l+1
1085 IF i$="z" THEN  LET k=k-(k>0)
1090 IF i$="x" THEN  LET k=k+(k<31)
1095 PRINT AT l,k;"C": BEEP .002,1
1100 IF SCREEN$ (l+1,k)<>" " THEN  GO SUB 1
170
1105 IF z<>1 THEN  GO TO 1135
1110 PRINT AT i,q;" ": LET i=i+1
1115 IF i$="m" THEN  LET q=q+(q<28)
1120 IF i$="n" THEN  LET q=q-(q>3)
1125 PRINT AT i,q;"D": BEEP .002,-5
```

```
1130 IF SCREEN$ (i+1,q)<>" " THEN  GO SUB 1
190
1135 IF l=21 THEN  PRINT AT l,k;" ": LET t=
0
1140 IF i=21 THEN  LET s=s+b*2: BEEP .1,20:
 PRINT AT i,q;" ": LET i=0: LET z=0
1145 LET a=a+c
1150 PRINT AT b,a; INK RND*3; PAPER 7;"A"
1155 IF a<3 OR a>27 THEN  LET c=-c: LET b=b
-1
1160 IF b=3 THEN  GO TO 1260
1165 GO TO 1040
1170 PRINT AT l,k;" ": PRINT AT l+1,k-1;"
"
1175 LET t=0: IF l<20 THEN  PRINT AT l+2,k-
1;"   "
1180 IF i=l+1 OR i=l AND q=k AND z=1 THEN
GO SUB 1190
1185 LET k=t: LET l=k: RETURN
1190 PRINT AT i,q; FLASH 1;"*": LET li=li-1
: BEEP .1,-30: BEEP .1,-25
1195 LET z=0: PRINT AT i,q;" ";AT i+1,q-1;"
   ": IF li=0 THEN  GO TO 1260
1200 LET i=z: LET q=z: RETURN
1205 DATA 0,126,126,126,126,126,126,126
1210 DATA 0,24,110,219,255,110,24,0
1215 DATA 231,126,60,60,60,60,60,24
1220 DATA 28,28,8,62,8,8,20,34
1225 FOR a=USR "a" TO USR "d"+7: READ b: PO
KE a,b: NEXT a
1230 LET a=27: LET b=20
1235 LET c=-.6: LET t=0: LET l=1: LET z=0:
LET i=1
1240 LET x=0: LET f=1
1245 LET s=0
1250 LET li=10
1255 RETURN
1260 IF li=0 THEN  PRINT AT 0,30;"0"
1265 FOR o=-30 TO 30 STEP 2
1270 BEEP .01,o
1275 NEXT o
1280 PRINT  INK 0; FLASH 1;AT 15,9;"GIOCHI
ANCORA ?:"
```

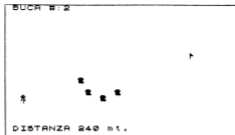
```
1285 IF s>hs THEN  LET hs=s
1290 INPUT a$
1295 IF a$(1)="s" THEN  GO SUB 1230: CLS :
PAUSE 100: GO TO 1035
1300 STOP
1305 CLS : PRINT AT 0,10; PAPER 0; INK 5; F
LASH 1;"WALL - RUN !"
1310 PRINT AT 3,0;"Devi cercare di far ragg
iungere"
1315 PRINT "ai tuoi uomini la Terra,mentre
"
1320 PRINT "gli Alieni invisibili costruisc
o-no un muro micidiale che puoi  distrugger
e solo con le bombe."
1325 PRINT "Fai in fretta prima che il temp
o"
1330 PRINT "finisca...!"
1335 PRINT : PRINT "Usa:"
1340 PRINT : PRINT "<Z> <X> per sganciare l
e Bombe"
1345 PRINT : PRINT "<N> <M> per sganciare g
li Uomini"
1350 PRINT : PRINT  FLASH 1;"BUONA FORTUNA"
1355 PRINT : PRINT "BATTI UN TASTO PER INIZ
IARE...."
1360 PAUSE 0: CLS : RETURN
```

GOLF

Questo programma simula il gioco del golf a nove buche ed è molto semplice da giocare. Il computer chiederà di volta in volta il numero della mazza che si intende usare. Lo scopo è quello di raggiungere la nona buca col minor numero di colpi. Si utilizza un metodo abbastanza insolito per definire i tre caratteri grafici utilizzati mediante l'istruzione BIN.

Questo metodo è pratico solo nel caso si voglia definire un numero piuttosto piccolo di caratteri, ma ha il vantaggio di mettere in risalto, con buona approssimazione, la forma che i caratteri avranno.

BUCA #:2

DISTANZA 240 mt.

```
1000 GO SUB 1970
1005 REM CHAMPIONSHIP GOLF
1010 REM
1015 REM DEFINIZIONE CARATTERI
1020 REM
1025 REM giocatore
1030 REM
1035 POKE USR "a"+0,BIN 00011000
1040 POKE USR "a"+1,BIN 00111100
1045 POKE USR "a"+2,BIN 01011010
1050 POKE USR "a"+3,BIN 00111100
1055 POKE USR "a"+4,BIN 00011000
1060 POKE USR "a"+5,BIN 00100100
1065 POKE USR "a"+6,BIN 00100100
1070 POKE USR "a"+7,BIN 01000010
1075 REM
1080 REM
1085 REM bandiera segna-buca
1090 REM
1095 POKE USR "b"+0,BIN 00010000
1100 POKE USR "b"+1,BIN 00011000
1105 POKE USR "b"+2,BIN 00011100
1110 POKE USR "b"+3,BIN 00011110
1115 POKE USR "b"+4,BIN 00010000
1120 POKE USR "b"+5,BIN 00010000
1125 POKE USR "b"+6,BIN 00010000
1130 POKE USR "b"+7,BIN 00010000
1135 REM
1140 REM bunkers in sabbia
```

```
1145 REM
1150 POKE USR "c"+0,BIN 00101000
1155 POKE USR "c"+1,BIN 01111100
1160 POKE USR "c"+2,BIN 11111110
1165 POKE USR "c"+3,BIN 01111100
1170 POKE USR "c"+4,BIN 01111000
1175 POKE USR "c"+5,BIN 01111100
1180 POKE USR "c"+6,BIN 01111110
1185 POKE USR "c"+7,BIN 00111110
1190 REM
1195 REM INIZIALIZZAZIONI
1200 REM
1205 DIM t(9): LET f=1
1210 LET hx=0: LET cx=0
1215 LET hy=0: LET th=0
1220 LET cy=0: PAPER 4: INK 7
1225 POKE 23609,50
1230 REM
1235 REM CORPO PRINCIPALE
1240 REM
1245 LET n=9: REM num.buche
1250 BORDER 4
1255 FOR j=1 TO n
1260 CLS
1265 PRINT AT 0,1; INK 0;"BUCA #:";j
1270 FOR e=1 TO 5
1275 LET bx=RND*5+10
1280 LET by=RND*5+10
1285 PRINT AT by,bx; PAPER 8; INK 6;"C"
1290 NEXT e
1295 LET x=INT (RND*5)+1
1300 LET y=INT (RND*5+12)
1305 LET tx=INT (RND*15+15)
1310 LET ty=INT (RND*10)+1
1315 LET d=SGN (tx-x)*SQR ((tx-x)*(tx-x)+(
ty-y)*(ty-y))
1320 PRINT AT ty,tx; INK 7;"B"
1325 PRINT AT y,x; PAPER 8; INK 0;"A"
1330 PRINT AT 20,1; INK 0;"DISTANZA ";INT
D*10;" mt."
1335 INPUT  INK 0;"CHE MAZZA SCEGLI (1-8)
?:";m
1340 IF m<1 OR m>8 THEN  GO TO 1335
```

```
1345 LET m=INT ((9-m)/f)+1
1350 GO SUB 1700
1355 GO SUB 1500
1360 LET f=1
1365 LET d=SGN (tx-thx)*SQR ((tx-thx)*(tx-
thx)+(ty-thy)*(ty-thy))
1370 IF d<-3 THEN  PRINT AT 20,1; INK 0;"F
UORI CAMPO-RIPROVA LA BUCA!": PAUSE 100: G
O TO 1260
1375 IF d<3 THEN  PRINT AT 20,1; INK 0;"SE
I SUL GREEN!  ": PAUSE 100: GO TO 1775
1380 PRINT AT y,x;" "
1385 LET x=thx
1390 LET y=thy
1395 GO TO 1320
1400 PRINT AT 2,3; INK 0;"HAI IMPIEGATO ";
t(j);" COLPI"
1405 PRINT AT 4,3; INK 0;"PER COMPLETARE L
A BUCA ";j
1410 PAUSE 100
1415 NEXT j
1420 REM
1425 REM RISULTATI
1430 REM
1435 BORDER 2: PAPER 1: INK 7
1440 CLS
1445 PRINT AT 2,4; INVERSE 1;"RISULTATI DE
LLA PARTITA": INVERSE 0
1450 PRINT
1455 FOR r=1 TO n
1460 PRINT TAB 8;"Buca ";r;
1465 IF t(r)=-1 THEN  PRINT " palla persa"
: GO TO 1475
1470 PRINT " ";t(r);" colpi"
1475 NEXT r
1480 INPUT "ALTRA PARTITA ?:";p$
1485 IF p$="s" THEN  RUN
1490 PAPER 7: BORDER 7: INK 0: STOP
1495 REM
1500 REM ROUTINE LANCIO
1505 REM
1510 PLOT  INK 8; PAPER 8; INVERSE 1;hx+cx
,hy+th+cy
```

```
1515 LET vt=(m*(2+RND*.3))
1520 LET th=0
1525 LET hx=0
1530 REM
1535 REM GESTIONE PALLINA
1540 REM
1545 LET q=(y-ty)/(tx-x)
1550 LET vv=vt*(SIN (45*PI/180))
1555 LET cx=(x+1)*8
1560 LET cy=21-y: LET cy=cy*8
1565 LET vh=vt*(COS (45*PI/180))
1570 LET th=th+vv
1575 LET hy=q*hx
1580 LET vv=vv-2.5
1585 LET hx=hx+vh
1590 LET hy=q*hx
1595 IF hx+cx>255 THEN  GO TO 1965
1600 IF hy+th+cy>175 THEN  GO TO 1965
1605 IF th<=0 THEN  GO TO 1630
1610 PLOT  PAPER 8; OVER 1; INK 8;hx+cx,hy
+th+cy
1615 PAUSE 2
1620 PLOT  PAPER 8; OVER 1; INK 8;hx+cx,hy
+th+cy
1625 GO TO 1570
1630 LET thx=INT ((hx+cx)/8)
1635 LET thy=INT ((175-hy-th-cy)/8)
1640 IF POINT (hx+cx,hy+th+cy)=1 THEN  GO
TO 2045
1645 PLOT  INK 8; PAPER 8;hx+cx,hy+th+cy
1650 RETURN
1655 REM
1660 REM PALLINA PERSA
1665 REM
1670 PRINT AT 2,1; INK 2; PAPER 7; INVERSE
 1;"HAI PERSO LA PALLA"
1675 BEEP 1,0
1680 LET t(j)=-1
1685 PAUSE 75
1690 GO TO 1415
1695 REM
1700 REM ROUTINE SWING
1705 REM
```

```
1710 LET t(j)=t(j)+1
1715 LET sx=x*8+4
1720 LET sy=21-y: LET sy=sy*8+4
1725 FOR k=-5 TO -40 STEP -5
1730 LET a=k/30*PI
1735 LET xs=7*SIN a: LET ys=7*COS a
1740 PLOT  INK 8; PAPER 8;sx,sy: DRAW  INK
 8; PAPER 8; OVER 1;xs,ys
1745 IF k<>-30 THEN  PAUSE 5
1750 IF k=-30 THEN  BEEP .1,-2
1755 PLOT  INK 8; PAPER 8;sx,sy: DRAW  INK
 8; PAPER 8; OVER 1;xs,ys
1760 NEXT k
1765 RETURN
1770 REM
1775 REM SUL GREEN
1780 REM
1785 PAUSE 10: PAPER 4: CLS
1790 LET gx=INT (RND*5)+1
1795 LET gy=15
1800 LET hx=INT (RND*15)+10
1805 LET hy=15
1810 LET d=hx-gx
1815 IF d<0 THEN  LET d=ABS (d)
1820 PRINT AT hy,hx; INK 0;"O"
1825 PRINT AT gy,gx; INK 0;"A"
1830 PRINT AT 18,1; INK 0;"GREEN - BUCA A
 ";d;" mt."
1835 INPUT "CHE MAZZA SCEGLI (1-8) ?:";m
1840 IF m<1 OR m>8 THEN  GO TO 1835
1845 LET t(j)=t(j)+1
1850 LET h1=9-m+INT (RND*2)
1855 LET d=d-h1
1860 FOR w=gx+1 TO gx+h1
1865 PRINT AT hy,w; INK 0;"."
1870 PAUSE 10
1875 PRINT AT hy,w;" "
1880 NEXT w
1885 PRINT AT gy,gx;" "
1890 LET gx=gx+h1
1895 IF gx=hx THEN  GO TO 1925
1900 IF d<0 THEN  CLS : LET gx=hx+d: GO TO
 1810
```

```
1905 GO TO 1820
1910 REM
1915 REM PALLA IN BUCA
1920 REM
1925 CLS : GO TO 1400
1930 REM
1935 REM PALLA NEL BUNKER
1940 REM
1945 PRINT AT 20,1; INK 0;"SEI FINITO DRIT
TO NEL BUNKER!"
1950 PAUSE 100
1955 LET f=2
1960 GO TO 1400
1965 LET hy=0: LET ty=hy: LET cy=hy: LET h
x=hy: LET cx=hy: GO TO 1660
1970 REM ----------------
1975 REM  TITOLO DI TESTA
1980 REM ----------------
1985 BORDER 1: PAPER 1: INK 7: CLS
1990 PRINT AT 2,7; FLASH 1;"CHAMPIONSHIP G
OLF"
1995 PRINT : PRINT "BENVENUTO AL MIGLIOR G
OLF A 9"
2000 PRINT "BUCHE DEL MONDO!"
2005 PRINT "DEVI USARE TUTTO IL TUO FIUTO"
2010 PRINT "PER DOSARE LA FORZA DEI COLPI
E"
2015 PRINT "ANDARE IN BUCA COL MINOR NUMER
O"
2020 PRINT "POSSIBILE DI TENTATIVI."
2025 PRINT : PRINT "SCEGLI LE MAZZE DELLA
TUA SACCA"
2030 PRINT "CON GRANDE CURA!"
2035 PRINT : PRINT : PRINT "PREMI UN TASTO
 PER INIZIARE "
2040 PAUSE 0: RETURN
```

RENNE

È un programma ideato nel periodo natalizio ma giocabile anche a Pasqua tanto è divertente.

Babbo Natale vola con la sua slitta sopra i tetti di alcune case. Deve raccogliere i regali che piovono dal cielo e calarli nei camini delle case ma non deve farsi vedere dai bambini che in queste dormono altrimenti il gioco finisce. Il tutto diventa più difficile man mano che si superano i primi livelli di gioco.

REGALI: 16   RECORD: 53

```
1000 PAPER 1: INK 7: BORDER 1: CLS
1005 POKE 23658,8: GO SUB 1230
1010 LET hs=0
1015 GO SUB 1400
1020 LET sk=.98: LET s=0
1025 LET a$="A": LET b$="B"
1030 LET c$="C": LET x=27
1035 DIM d$(28)
1040 LET g=1: LET h=INT (RND*28)+2
1045 PRINT AT 0,22;"RECORD:";hs
1050 LET p=0
1055 PRINT AT 0,0;"REGALI:";s
1060 IF INKEY$="X" THEN  LET x=x+(2 AND
x<27)
1065 LET a$="D": LET b$="E": LET c$="F"
1070 IF INKEY$="Z" THEN  LET x=x-(2 AND
x>1)
1075 LET a$="A": LET b$="B": LET c$="C"
1080 PRINT AT 10,x-2;"       ";AT 10,x;
INK 2;a$; INK 0;b$; INK 2;c$;
1085 IF NOT p THEN  LET g=g+1: PRINT AT
g-1,h;" ";AT g,h; INK 6;"G"
```

```
1090 IF g=10 THEN  IF h=x+2 OR h=x+1 OR
h=x THEN  LET p=1
1095 IF g=12 THEN  IF SCREEN$ (g+1,h)=":
" THEN  GO TO 1120
1100 IF g=12 THEN  PRINT AT 12,h;" ": LE
T g=1: LET h=INT (RND*28)+2
1105 IF p THEN  IF INKEY$="M" THEN  LET
g=10: LET h=x-1+(a$="D")+(3 AND c$="C"):
 LET p=0
1110 IF RND>sk THEN  LET q=INT (RND*7)+1
: LET e=INT (RND*4): PRINT  PAPER 3;AT 1
6,q*4-e+1;" ": LET d$(q*4-e)="p": IF d$(
q*4-3 TO q*4)="pppp" THEN  GO TO 1155
1115 GO TO 1060
1120 PRINT AT 12,h;" "
1125 IF SCREEN$ (18,h)="_" THEN  GO TO 1
040
1130 LET s=s+1: PRINT AT 0,0;"REGALI:";s
1135 PRINT AT 18,h; INK 0; PAPER 6;"__";
AT 19,h;"__"
1140 FOR f=1 TO 3: BEEP .001,30: NEXT f
1145 FOR f=4 TO 28 STEP 4: IF SCREEN$ (1
8,f)="_" THEN  NEXT f: LET sk=sk-.02: CL
S : GO SUB 1400: GO TO 1025
1150 GO TO 1040
1155 IF SCREEN$ (18,q*4)<>"_" THEN  GO T
O 1180
1160 GO TO 1060
1165 RESTORE 1170: LET z=.3: LET c=.6: F
OR f=1 TO 26: READ a,b: BEEP a,b: NEXT f
1170 DATA z,6,z,6,c,6,z,6,z,6,c,6,z,6,z,
9,z,2,z,4,1,6,z,6,z,7,z,7,z,7,z,7,z,7,z,
6,z,6,z,6,z,9,z,9,z,7,z,4,c,2,c,2
1175 RETURN
1180 FOR f=10 TO 1 STEP -.5: PRINT AT f,
x; INK 2;a$; INK 0;b$; INK 2;c$;AT f+1,x
;"    ": BEEP .005,2*f: NEXT f
1185 IF s>hs THEN  LET hs=s
1190 PRINT AT 1,x;"      "
1195 PRINT AT g,h;" "
1200 PRINT  FLASH 1;AT 18,q*4-1; INK 0;
PAPER 6;"__";AT 19,q*4-1;"K_"
1205 PRINT AT 8,8; FLASH 1;"TI HANNO VIS
```

```
TO !"
1210 GO SUB 1165
1215 PRINT AT 11,2; FLASH 1;"PREMI UN TA
STO PER GIOCARE.."
1220 PAUSE 0: PAUSE 0
1225 CLS : GO TO 1015
1230 CLS : PRINT AT 0,9; FLASH 1;"RENNE
& REGALI"
1235 PRINT : PRINT "E'Natale e Tu,che se
i il Grande"
1240 PRINT "BABBO NATALE,devi consegnare
 al"
1245 PRINT "piu'presto tutti i REGALI ch
e "
1250 PRINT "piovono dal tuo deposito su
nel"
1255 PRINT "cielo.Ma ATTENTO! devi fare
in "
1260 PRINT "modo che i bambini non ti ve
dano"
1265 PRINT "altrimenti non potrai piu' f
are"
1270 PRINT "i tuoi magnifici Regali."
1275 PRINT "La NEVE sui tetti si sciogli
e  "
1280 PRINT "perche'i bambini che hanno r
ice-"
1285 PRINT "vuto i Regali accendono la l
uce"
1290 PRINT "Se ti scoprono,quando la Nev
e si"
1295 PRINT "e'sciolta hai perso."
1300 PRINT "--------------------
-----"
1305 PRINT "<Z> per andare a SINISTRA"
1310 PRINT "<X> per andare a DESTRA"
1315 PRINT "<M> per calare i regali  nei
"
1320 PRINT "     camini delle Casette"
1325 PRINT "--------------------
-----"
1330 PRINT  FLASH 1;"BATTI UN TASTO PER
GIOCARE....."
```

```
1335 RESTORE 1340: FOR y=USR "a" TO USR
"k"+7: READ x: POKE y,x: NEXT y
1340 DATA 0,0,1,0,0,0,0,0
1345 DATA 192,64,160,224,127,126,99,82
1350 DATA 8,28,8,124,156,72,63,128
1355 DATA 16,56,16,62,56,18,252,1
1360 DATA 3,2,5,7,254,126,198,74
1365 DATA 0,0,128,0,0,0,0,0
1370 DATA 0,0,0,54,54,0,54,54
1375 DATA 0,1,3,7,15,31,63,127
1380 DATA 0,128,192,224,240,248,252,254
1385 DATA 255,220,73,65,64,0,0,0
1390 DATA 24,60,86,60,24,126,255,255
1395 PAUSE 0: CLS : RETURN
1400 FOR x=2 TO 26 STEP 4
1405 PRINT  INK 3;AT 13,x+1;":"; PAPER 3
; INK 5;AT 14,x+1;"H"; PAPER 1;"I";AT 15
,x;"H■I"; INK 7; PAPER 3;AT 16,x;"JJJJ"
;AT 17,x; PAPER 1; INK 3;"■■■■";AT 18,x;
"■  ■";AT 19,x;"■  ■";AT 20,x;"■■■■";AT
21,x;"■■■■"
1410 NEXT x
1415 OVER 1: INK 3: FOR x=48 TO 208 STEP
 32
1420 PLOT x,0: DRAW 0,39: NEXT x: INK 7:
 OVER 0:
1425 FOR x=24 TO 216 STEP 32: PLOT x,23:
 DRAW 15,0: NEXT x
1430 RETURN
```

THE WALL

Questo programma è simile a quello che un po' di tempo fa era uno dei videogame più presenti nei bar e nelle sale gioco.

Lo scopo del gioco è di abbattere un muro di mattoncini posti nella parte alta dello schermo e nel frattempo proteggere la fila di mattoncini posti in basso.

Ogni mattoncino abbattuto vale 10 punti mentre quelli in basso non modificano il punteggio. Ogni 1000 vengono aggiunti 100 punti. Il gioco finisce quando i mattoncini in basso sono esauriti, mentre quando i mattoncini in alto sono stati tutti abbattuti, il gioco riprende con la ricostruzione del muro lasciando la fila in basso inalterata.

In qualsiasi momento si può interrompere il gioco premendo i tasti SHIFT-BREAK.

Dando il comando RUN si dovrà attendere una lunga pausa necessaria per memorizzare il linguaggio macchina, verrà quindi chiesto il grado di difficoltà (da 1 a 4).

I comandi per muovere la racchetta sono:

1——— a sinistra lentamente
5——— a destra lentamente
1–2–– a sinistra velocemente
4–5–– a destra velocemente

In questi ultimi due casi i tasti vanno premuti contemporaneamente, mentre premendo solo il 2 o solo il 4, la pallina aumenterà la sua velocità.

Il programma si divide in due parti:

– una parte molto consistente costituita dal L/M;
– una parte in Basic che gestisce la memorizzazione del L/M, dei punteggi, le varie richieste e la costruzione del campo di gioco.

Per quanto riguarda la prima consiglio solo una cosa: molta pazienza nel digitare tutti i codici.

Per la seconda avverto che è possibile qualche modifica. Ad esempio la disposizione del muro e quindi il numero di mattoncini.

La posizione del muro e le coordinate iniziali della pallina (POKE 32255,Y, POKE 32256,X) sono strettamente legate.

Queste coordinate devono essere in genere, una combinazione di numeri pari o dispari. Suggerisco al lettore di fare diverse prove in funzione del muro che si vuole costruire.

Ultima cosa: il numero di mattoni N deve essere memorizzato con POKE 31934,N.

```
1000 REM
1005 DATA   31600,31688
1010 REM
1015 DATA "CD","DD","7E","3E","20","B9"
1020 DATA "C8","3E","91","B9","20","4D"
1025 DATA "3A","FA","7D","FE","00","20"
1030 DATA "09","32","FC","7D","3C","32"
1035 DATA "FA","7D","18","0E","32","FC"
1040 DATA "7D","3D","32","FA","7D","18"
1045 DATA "05","52","69","67","6F","52"
1050 DATA "3E","20","D7","3E","20","D7"
1055 DATA "3E","08","D7","3E","08","D7"
1060 DATA "3A","FF","7D","FE","14","20"
1065 DATA "11","3A","DB","7E","3C","32"
1070 DATA "DB","7E","21","00","01","11"
1075 DATA "1E","00","CD","B5","03","C9"
1080 DATA "CD","27","7D","3A","DC","7E"
1085 DATA "3C","32","DC","7E","C9"
1090 REM
1095 DATA   31689,31795
1100 REM
1105 DATA "CD","B7","7E","FE","00","28"
1110 DATA "12","3A","FF","7D","3C","32"
1115 DATA "FF","7D","3E","00","32","FA"
1120 DATA "7D","3C","32","FC","7D","18"
1125 DATA "10","3A","FF","7D","3D","32"
1130 DATA "FF","7D","3E","00","32","FC"
1135 DATA "7D","3C","32","FA","7D","3A"
1140 DATA "FB","7D","FE","01","20","09"
1145 DATA "3A","00","7E","3C","32","00"
1150 DATA "7E","18","07","3A","00","7E"
```

```
1155 DATA "3D","32","00","7E","3E","93"
1160 DATA "B9","20","09","3E","01","32"
1165 DATA "FB","7D","3D","32","FD","7D"
1170 DATA "3E","94","B9","20","09","3E"
1175 DATA "01","32","FD","7D","3D","32"
1180 DATA "FB","7D","3E","02","CD","01"
1185 DATA "16","01","03","00","11","FE"
1190 DATA "7D","CD","3C","20","C9"
1195 REM
1200 DATA   31796,31890
1205 REM
1210 DATA "00","00","00","00","16","0B"
1215 DATA "0B","20","93","8F","8F","8F"
1220 DATA "94","52","01","FE","F7","ED"
1225 DATA "78","FE","FF","20","05","06"
1230 DATA "00","10","FE","C9","FE","FE"
1235 DATA "28","10","FE","FC","20","19"
1240 DATA "CD","8D","7D","3A","3A","7C"
1245 DATA "A7","C8","3D","32","3A","7C"
1250 DATA "CD","8D","7D","3A","3A","7C"
1255 DATA "A7","C8","3D","32","3A","7C"
1260 DATA "C9","FE","EF","28","10","FE"
1265 DATA "E7","C0","CD","98","7D","3A"
1270 DATA "3A","7C","FE","1A","C8","3C"
1275 DATA "32","3A","7C","CD","98","7D"
1280 DATA "3A","3A","7C","FE","1A","C8"
1285 DATA "3C","32","3A","7C","C9"
1290 REM
1295 DATA   31906,31998
1300 REM
1305 DATA "CD","01","7E","06","01","76"
1310 DATA "10","FD","CD","54","1F","D2"
1315 DATA "E5","7C","CD","42","7C","3A"
1320 DATA "DB","7E","FE","10","28","2B"
1325 DATA "3A","DC","7E","FE","5A","20"
1330 DATA "E1","AF","32","FC","7D","32"
1335 DATA "FD","7D","3C","32","FA","7D"
1340 DATA "32","FB","7D","3E","0A","21"
1345 DATA "00","06","11","05","00","E5"
1350 DATA "CD","B5","03","E1","11","08"
1355 DATA "00","A7","ED","52","20","F0"
1360 DATA "C9","06","4B","C5","21","00"
1365 DATA "01","11","01","00","E5","CD"
```

```
1370 DATA "B5","03","E1","11","10","00"
1375 DATA "A7","ED","52","20","F0","C1"
1380 DATA "10","E9","C9"
1385 REM
1390 DATA  32000,32085
1395 REM
1400 DATA "16","15","00","14","01","50"
1405 DATA "55","4E","54","49","3A","30"
1410 DATA "30","30","30","30","93","94"
1415 DATA "47","52","41","44","4F","3A"
1420 DATA "31","20","4D","41","58","3A"
1425 DATA "30","30","30","30","30","93"
1430 DATA "94","14","00","CD","AA","7E"
1435 DATA "3C","FE","3A","20","07","3E"
1440 DATA "30","32","0E","7D","18","04"
1445 DATA "32","0E","7D","C9","3A","0D"
1450 DATA "7D","3C","FE","3A","20","07"
1455 DATA "3E","30","32","0D","7D","18"
1460 DATA "04","32","0D","7D","C9","21"
1465 DATA "0F","00","11","28","00","E5"
1470 DATA "CD","B5"
1475 REM
1480 DATA  32086,32180
1485 REM
1490 DATA "03","E1","11","10","00","A7"
1495 DATA "ED","5A","7D","FE","FF","20"
1500 DATA "ED","3E","31","32","0D","7D"
1505 DATA "3A","0C","7D","3C","FE","3A"
1510 DATA "20","07","3E","30","32","0C"
1515 DATA "7D","18","04","32","0C","7D"
1520 DATA "C9","3A","0B","7D","3C","FE"
1525 DATA "3A","20","06","3E","30","32"
1530 DATA "0B","7D","C9","32","0B","7D"
1535 DATA "C9","2A","39","7C","22","A2"
1540 DATA "7D","11","A1","7D","18","03"
1545 DATA "11","38","7C","01","09","00"
1550 DATA "C3","3C","20","16","12","00"
1555 DATA "93","8F","8F","8F","94","20"
1560 DATA "3E","02","CD","01","16","CD"
1565 DATA "98","7D","C3","C1","7C"
1570 REM
1575 DATA  32250,32350
1580 REM
```

```
1585 DATA "01","00","00","01","16","13"
1590 DATA "0F","3E","02","CD","01","16"
1595 DATA "01","27","00","11","00","7D"
1600 DATA "CD","3C","20","01","03","00"
1605 DATA "11","FE","7D","CD","3C","20"
1610 DATA "3E","20","D7","2A","FF","7D"
1615 DATA "ED","4B","FA","7D","ED","42"
1620 DATA "ED","4B","FC","7D","ED","4A"
1625 DATA "22","FF","7D","00","00","00"
1630 DATA "00","00","00","00","00","00"
1635 DATA "01","03","00","11","FE","7D"
1640 DATA "CD","3C","20","CD","70","7B"
1645 DATA "3E","90","D7","3A","00","7E"
1650 DATA "FE","00","20","12","3E","00"
1655 DATA "32","FB","7D","3C","32","FD"
1660 DATA "7D","21","0F","02","11","0A"
1665 DATA "00","CD","B5","03","3A"
1670 REM
1675 DATA   32351,32451
1680 REM
1685 DATA "00","7E","FE","1F","20","12"
1690 DATA "21","00","02","11","0A","00"
1695 DATA "CD","B5","03","3E","00","32"
1700 DATA "FD","7D","3C","32","FB","7D"
1705 DATA "3A","FF","7D","FE","00","20"
1710 DATA "12","21","00","02","11","0A"
1715 DATA "00","CD","B5","03","3E","00"
1720 DATA "32","FA","7D","3C","32","FC"
1725 DATA "7D","3A","FF","7D","FE","14"
1730 DATA "20","12","21","00","02","11"
1735 DATA "0A","00","CD","B5","03","3E"
1740 DATA "00","32","FC","7D","3C","32"
1745 DATA "FA","7D","C9","21","00","03"
1750 DATA "11","10","00","CD","B5","03"
1755 DATA "3A","0E","7D","C9","21","00"
1760 DATA "05","11","05","00","CD","B5"
1765 DATA "03","3A","FA","7D","C9"
1770 REM
1775 DATA   32475,32599
1780 REM
1785 DATA "10","2B","2A","36","5C","24"
1790 DATA "11","80","20","CD","1B","7F"
1795 DATA "14","1E","90","21","3A","7F"
```

```
1800 DATA "D5","ED","5B","84","5C","0E"
1805 DATA "02","06","04","1A","BE","20"
1810 DATA "0C","14","10","F9","23","0D"
1815 DATA "20","F3","D1","06","00","4A"
1820 DATA "C9","06","00","09","D1","14"
1825 DATA "7B","BA","20","DE","2A","7B"
1830 DATA "5C","1E","A5","CD","1B","7F"
1835 DATA "01","00","00","C9","01","08"
1840 DATA "00","D5","ED","5B","84","5C"
1845 DATA "1A","14","BE","20","0A","23"
1850 DATA "0D","20","F7","D1","06","00"
1855 DATA "4A","D1","C9","09","D1","14"
1860 DATA "7B","BA","20","E2","C9","0F"
1865 DATA "00","F0","00","FF","00","00"
1870 DATA "0F","0F","0F","F0","0F","FF"
1875 DATA "0F","00","F0","0F","F0","F0"
1880 DATA "F0","FF","F0","00","FF","0F"
1885 DATA "FF","F0","FF","FF","FF"
1890 REM
1895 DATA  USR "a",USR "e"+7
1900 REM
1905 DATA "00","18","3C","7E","7E","3C"
1910 DATA "18","00","FF","80","80","80"
1915 DATA "80","80","80","FF","FF","01"
1920 DATA "01","01","01","01","01","FF"
1925 DATA "00","01","03","07","07","03"
1930 DATA "01","00","00","80","C0","E0"
1935 DATA "E0","C0","80","00"



9000 REM
9005 REM   FINE   DATA WALL
9010 REM
9015 CLEAR 31599: REM RAMTOP LOW
9020 PRINT AT 12,7; FLASH 1;"Attendi un
attimo!"
9025 FOR v=0 TO 9
9030 READ i: READ f
9035 PRINT AT 14,19; INVERSE 1;f
9040 FOR p=i TO f: READ a$
9045 LET h=CODE a$(1)-48
9050 LET h=h-7*(a$(1)>"@")
```

```
9055 LET l=CODE a$(2)-48
9060 LET l=l-7*(a$(2)>"@")
9065 PRINT AT 14,8; FLASH 1;p;AT 14,15;
INVERSE 1;a$
9070 POKE p,16*h+l: BEEP .01,20
9075 NEXT p: BEEP .01,10: NEXT v
9080 REM
9085 REM       THE WALL
9090 REM
9095 CLEAR : GO TO 9250
9100 LET a$=CHR$ 145+CHR$ 146
9105 PAPER 1: INK 9: CLS
9110 FOR a=0 TO 31 STEP 4
9115 PRINT  PAPER 2; FLASH 1;AT 20,a;a$
9120 PRINT  PAPER 6; FLASH 1;AT 20,a+2;a
$
9125 NEXT a
9130 FOR g=0 TO 2
9135 FOR a=g TO 31-g STEP 2
9140 PRINT  PAPER g+1;AT g,a;a$
9145 PRINT  PAPER g+1;AT 8-g,a;a$
9150 NEXT a: NEXT g
9155 RANDOMIZE USR 32170
9160 RANDOMIZE USR 31906
9165 POKE 32255,19: POKE 32256,15
9170 IF PEEK 32476=90 THEN  POKE 32476,0
: GO TO 9130
9175 BORDER 1: FLASH 1
9180 PRINT AT 9,8;"FINE ";
9185 PRINT CHR$ 145+CHR$ 146;
9190 PRINT " GIOCO "+CHR$ 144
9195 LET a$="": LET b$=a$
9200 FOR a=32011 TO 32015
9205 LET a$=a$+CHR$ PEEK a
9210 LET b$=b$+CHR$ PEEK (a+19)
9215 POKE a,48: NEXT a
9220 LET a=VAL a$: LET b=VAL b$
9225 IF a<=b THEN  GO TO 9250
9230 FOR a=1 TO 5
9235 POKE (32029+a),CODE a$(a)
9240 NEXT a
9245 PRINT AT 11,3;"Hai il punteggio piu
'alto"
```

```
9250 FLASH 0: INPUT "Difficolta' <1-4> "
; LINE a$
9255 IF a$="" THEN  GO TO 9250
9260 LET a=VAL a$
9265 IF a>4 OR a<1 THEN  GO TO 9250
9270 POKE 31910,5-a: POKE 32024,a+48
9275 POKE 31801,20-a*2.5
9280 CLEAR : PRINT AT 8,8;"Premi un tast
o"
9285 PAUSE 0: BORDER 4
9290 POKE 32475,0: POKE 32476,0
9295 POKE 32255,19: POKE 32256,15
9300 RUN 9100
```

PING

Questo gioco può essere la base per farne uno più complesso. Infatti assomiglia al più complicato THE WALL presentato in questo libro. È molto semplice sia come programmazione che come gioco. Può essere ulteriormente migliorato.

Lo scopo è quello di respingere la pallina con la racchetta che di volta in volta sale verso l'alto di una linea.

I tasti da usare sono 1 e 0 per muovere a sinistra e a destra.

```
1000 BORDER 1: PAPER 7: INK 0
1005 LET h=0: LET ht=0: LET d=19: LET bb=0
1010 GO SUB 1220: GO SUB 1185
1015 LET bb=bb+1
1020 IF bb>10 THEN  GO TO 1300
1025 LET a=1: LET b=a
1030 LET v=1: LET w=v
1035 LET x=10: LET y=d
1040 PRINT AT 21,5;"PALLA:";bb;
1045 PRINT AT y,x; PAPER 8; INK 0;" ███ "
1050 PRINT AT 21,19;"PUNTI:";ht
1055 GO SUB 1110: GO SUB 1090
1060 IF y<>b+w THEN  LET y=d: GO TO 1045
1065 BEEP 1,0
```

```
1070 PRINT AT y,x; PAPER 8;"     ": REM 5 s
pazi
1075 PRINT AT b,a; PAPER 8;" "
1080 LET d=d+(d<19)
1085 LET h=0: GO TO 1015
1090 LET a$=INKEY$
1095 IF a$="1" THEN  LET x=x-(x>0)
1100 IF a$="0" THEN  LET x=x+(x<27)
1105 RETURN
1110 PRINT AT b,a; PAPER 8; INK 8;" "
1115 LET a=a+v: LET b=b+w
1120 IF a=30 OR a=1 THEN  LET v=-v: BEEP .0
1,40
1125 IF b=1 THEN  LET w=-w: BEEP .01,-15
1130 IF y=b+w THEN  GO TO 1145
1135 PRINT AT b,a; PAPER 8;"A"
1140 RETURN
1145 LET r=a-x
1150 IF r<1 OR r>3 THEN  GO TO 1135
1155 LET w=-w: BEEP .01,5
1160 LET h=h+1: LET ht=ht+1
1165 IF h<>2 THEN  GO TO 1135
1170 LET h=0: LET d=d-1
1175 PRINT AT y,x; PAPER 8;"     ": REM 5 s
pazi
1180 GO TO 1110
1185 FOR i=0 TO 31
1190 PRINT AT 0,i; PAPER 0;" "
1195 NEXT i
1200 PRINT AT 0,4; INK 7; PAPER 0;"1 = Sini
stra  0 = Destra"
1205 FOR i=0 TO 20
1210 PRINT AT i,0; PAPER 0;" ";AT i,31;" "
1215 NEXT i: RETURN
1220 PRINT AT 7,0;
1225 LET c=2: GO SUB 1240
1230 LET c=6: GO SUB 1240
1235 GO TO 1255
1240 FOR i=1 TO 7*32
1245 PRINT  PAPER c;" ";
1150 IF r<1 OR r>3 THEN  GO TO 1135
1155 LET w=-w: BEEP .01,5
1160 LET h=h+1: LET ht=ht+1
```

```
1165 IF h<>2 THEN  GO TO 1135
1170 LET h=0: LET d=d-1
1175 PRINT AT y,x; PAPER 8;"     ": REM 5 s
pazi
1180 GO TO 1110
1185 FOR i=0 TO 31
1190 PRINT AT 0,i; PAPER 0;" "
1195 NEXT i
1200 PRINT AT 0,4; INK 7; PAPER 0;"1 = Sini
stra  0 = Destra"
1205 FOR i=0 TO 20
1210 PRINT AT i,0; PAPER 0;" ";AT i,31;" "
1215 NEXT i: RETURN
1220 PRINT AT 7,0;
1225 LET c=2: GO SUB 1240
1230 LET c=6: GO SUB 1240
1235 GO TO 1255
1240 FOR i=1 TO 7*32
1245 PRINT  PAPER c;" ";
1250 NEXT i: RETURN
1255 POKE USR "a"+0,BIN 00111100
1260 POKE USR "a"+1,BIN 01111110
1265 POKE USR "a"+2,BIN 11111111
1270 POKE USR "a"+3,BIN 11111111
1275 POKE USR "a"+4,BIN 11111111
1280 POKE USR "a"+5,BIN 11111111
1285 POKE USR "a"+6,BIN 01111110
1290 POKE USR "a"+7,BIN 00111100
1295 RETURN
1300 INK 0: PAPER 7: CLS
1305 PRINT AT 10,10;"Punteggio:";ht
1310 INPUT "Giochi ancora ?:";a$
1315 IF a$(1)="s" THEN  RUN
1320 BORDER 7: CLS : PRINT "Ciao"
```

GENERATORE MORSE

Il programma seguente è in grado di tradurre un messaggio scritto dall'utente generando una serie di BEEP secondo la codifica MORSE.

Le opzioni che offre il programma sono le seguenti:

1) possibilità di variare la velocità delle parole tradotte al minuto,
2) possibilità di generare parole casuali a scopo dimostrativo,
3) si può ripetere un messaggio un numero di volte infinito,
4) e ovviamente si può sentire all'istante la codifica del carattere battuto sulla tastiera.

Un avviso:

- non si usano caratteri grafici per cui, ad esempio alla linea 1160, bisogna introdurre le lettere tra apici come caratteri A.S.C.I.I. normali.

```
1000 BORDER 1: PAPER 1: LET a=2: LET b=6
: CLS
1005 FOR i=21 TO 0 STEP -1
1010 PRINT AT i,8; INK a; PAPER b; FLASH
 1;"GENERATORE-MORSE"
1015 LET c=a: LET a=b: LET b=c: NEXT i
1020 PRINT #0;AT 1,9;"PREMI UN TASTO": P
AUSE 0
1025 BORDER 5: PAPER 5: INK 0: CLS
1030 POKE 23658,8: RESTORE
1035 DIM A$(47,6)
1040 FOR I=1 TO 47
1045 READ A$(I): NEXT I
1050 DATA "131313","","111111","","33333
","13333","11333","11133","11113","11111
","31111","33111","33311","33331","33311
1","313131","","","","113311",""
1055 DATA "13","3111","3131","311","1","
1131","331","1111","11","1333","313","13
11","33","31","333","1331","3313","131",
"111","3","113","1113","133","3113","313
3","3311"
1060 LET E=12: LET F=.058
1065 CLS : PRINT AT 4,8;"GENERATORE MORS
E"
1070 PRINT AT 6,2;"PREMI UNO DEI SEGUENT
I TASTI"
1075 PRINT AT 8,1;"V = SELEZIONE VELOCIT
```

```
A'(MAX.20)"
1080 PRINT AT 10,1;"L = INPUT LETTERE"
1085 PRINT AT 12,1;"R = RIPETIZIONE MESS
AGGIO"
1090 PRINT AT 14,1;"C = PAROLE CASUALI"
1095 PRINT  FLASH 1;AT 17,0;"VELOCITA' =
 ";E;" PAROLE PER MINUTO"
1100 PRINT AT 21,10;"F PER FINIRE"
1105 PAUSE 10: GO SUB 1425
1110 IF G$="C" THEN  GO TO 1145
1115 IF G$="L" THEN  GO TO 1205
1120 IF G$="R" THEN  GO TO 1250
1125 IF G$="V" THEN  GO TO 1335
1130 IF G$="F" THEN  GO TO 1440
1135 IF G$="" THEN  GO TO 1105
1140 PRINT AT 19,11;"SBAGLIATO!": BEEP .
5,-40: GO TO 1065
1145 CLS : PRINT  BRIGHT 1;AT 0,9;"PAROL
E CASUALI"
1150 PRINT AT 1,5;"PREMI ENTER PER FINIR
E"
1155 PRINT AT 2,6;"UN TASTO PER FERMARE"
1160 LET X$="ABCDEFGHIJKLMNOPQRSTUVWXYZ.
,:;?1234567890"
1165 DIM H$(5)
1170 FOR X=1 TO 5
1175 LET H$(X)=X$(INT (RND*LEN X$)+1)
1180 NEXT X
1185 LET D$=H$
1190 GO SUB 1310
1195 PAUSE 5: IF INKEY$<>"" THEN  GO SUB
 1410
1200 GO TO 1170
1205 CLS : PRINT  BRIGHT 1;" CODIFICA MO
RSE : PREMI I TASTI.": PRINT
1210 PRINT AT 1,5;"PREMI ENTER PER FINIR
E": PRINT
1215 GO SUB 1415
1220 LET A=CODE G$-43
1225 IF A=-30 THEN  GO TO 1065
1230 IF A=-11 THEN  PRINT " ";: PAUSE 10
: GO TO 1215
1235 IF A=2 OR A=4 OR A=17 OR A=18 OR A=
```

```
19 OR A=21 OR A<1 OR A>47 THEN  BEEP .5,
-40: GO TO 1215
1240 PRINT G$;
1245 GO SUB 1365: GO TO 1215
1250 CLS
1255 LET ZZ=0: INPUT  BRIGHT 1;"DAMMI IL
 MESSAGGIO ";D$
1260 GO SUB 1285: IF ZZ THEN  GO TO 1255
1265 CLS : PRINT  BRIGHT 1;AT 1,5;"PREMI
 ENTER PER FINIRE"
1270 PRINT : PRINT
1275 GO SUB 1310
1280 PAUSE 10: GO TO 1275
1285 FOR K=1 TO LEN D$
1290 LET A=CODE D$(K)-43
1295 IF A=-11 THEN  GO TO 1305
1300 IF A=2 OR A=4 OR A=17 OR A=18 OR A=
19 OR A=21 OR A<1 OR A>47 THEN  PRINT AT
 0,6;D$(K);" CARATTERE INESATTO": LET ZZ
=1: BEEP .3,-40
1305 NEXT K: RETURN
1310 FOR K=1 TO LEN D$
1315 PRINT D$(K);
1320 LET A=CODE D$(K)-43
1325 GO SUB 1365: NEXT K
1330 PRINT : RETURN
1335 CLS
1340 INPUT "PAROLE PER MINUTO ? ";E
1345 IF E>20 THEN  PRINT AT 0,4;"SONO TR
OPPE...RIPROVA !!": GO TO 1340
1350 IF E<4 THEN  PRINT AT 0,4;"SONO POC
HE....RIPROVA !!": GO TO 1340
1355 LET F=.7/E
1360 GO TO 1065
1365 IF A=-11 THEN  PAUSE F*200: RETURN
1370 FOR I=1 TO 6
1375 IF A$(A,I)=" " THEN  GO TO 1395
1380 BEEP VAL A$(A,I)*F,40
1385 FOR Z=1 TO 70-E STEP E: NEXT Z
1390 NEXT I
1395 FOR Z=1 TO 140-E STEP E: NEXT Z
1400 IF INKEY$=CHR$ 13 THEN  GO TO 1065
1405 RETURN
```

```
1410 IF INKEY$<>"" THEN  GO TO 1410
1415 IF INKEY$="" THEN  GO TO 1415
1420 LET G$=INKEY$: RETURN
1425 OVER 1: PLOT 50,135: DRAW 0,9: DRAW
 156,0
1430 DRAW 0,-9: DRAW -156,0
1435 OVER 0: LET G$=INKEY$: RETURN
1440 POKE 23658,0
```

Una giusta via di mezzo tra la praticità di uso e la velocità di esecuzione di un programma è rappresentata dall'uso combinato del BASIC e del linguaggio macchina; la conoscenza di quest'ultimo è comunque indispensabile per poter sfruttare a pieno il proprio calcolatore.

Questa è la premessa da cui parte l'autore, per proporre una serie di interessanti programmi BASIC che si servono di routine scritte in linguaggio macchina.

La proposta è un po' diversa dal solito e per questo particolarmente interessante presentando un utilizzo intelligente e raffinato della macchina.

L. 15.000

Cod. 554D ISBN 88-7056-046-5